Journal Subline

LNCS 8360

Transactions on Computational Science XXII

Marina L. Gavrilova · C. J. Kenneth Tan
Editors-in-Chief

Springer

Marina L. Gavrilova C.J. Kenneth Tan (Eds.)

Transactions on Computational Science XXII

Springer

Editors-in-Chief

Marina L. Gavrilova
University of Calgary, AB, Canada
E-mail: mgavrilo@ucalgary.ca

C.J. Kenneth Tan
CloudFabriQ Ltd., London, UK
E-mail: cjtan@CloudFabriQ.com

ISSN 0302-9743 (LNCS) e-ISSN 1611-3349 (LNCS)
ISSN 1866-4733 (TCOMPSCIE) e-ISSN 1866-4741 (TCOMPSCIE)
ISBN 978-3-642-54211-4 e-ISBN 978-3-642-54212-1
DOI 10.1007/978-3-642-54212-1
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014930319

CR Subject Classification (1998): H.3, G.1, F.2, I.5, I.3, I.2

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

LNCS Transactions on Computational Science

Computational science, an emerging and increasingly vital field, is now widely recognized as an integral part of scientific and technical investigations, affecting researchers and practitioners in areas ranging from aerospace and automotive research to biochemistry, electronics, geosciences, mathematics, and physics. Computer systems research and the exploitation of applied research naturally complement each other. The increased complexity of many challenges in computational science demands the use of supercomputing, parallel processing, sophisticated algorithms, and advanced system software and architecture. It is therefore invaluable to have input by systems research experts in applied computational science research.

Transactions on Computational Science focuses on original high-quality research in the realm of computational science in parallel and distributed environments, also encompassing the underlying theoretical foundations and the applications of large-scale computation.

The journal offers practitioners and researchers the opportunity to share computational techniques and solutions in this area, to identify new issues, and to shape future directions for research, and it enables industrial users to apply leading-edge, large-scale, high-performance computational methods.

In addition to addressing various research and application issues, the journal aims to present material that is validated – crucial to the application and advancement of the research conducted in academic and industrial settings. In this spirit, the journal focuses on publications that present results and computational techniques that are verifiable.

Scope

The scope of the journal includes, but is not limited to, the following computational methods and applications:

- Aeronautics and Aerospace
- Astrophysics
- Big Data Analytics
- Bioinformatics
- Biometric Technologies
- Climate and Weather Modeling
- Communication and Data Networks
- Compilers and Operating Systems
- Computer Graphics
- Computational Biology
- Computational Chemistry

- Computational Finance and Econometrics
- Computational Fluid Dynamics
- Computational Geometry
- Computational Number Theory
- Data Representation and Storage
- Data Mining and Data Warehousing
- Information and Online Security
- Grid Computing
- Hardware/Software Co-design
- High-Performance Computing
- Image and Video Processing
- Information Systems
- Information Retrieval
- Modeling and Simulations
- Mobile Computing
- Numerical and Scientific Computing
- Parallel and Distributed Computing
- Robotics and Navigation
- Supercomputing
- System-on-Chip Design and Engineering
- Virtual Reality and Cyberworlds
- Visualization

Editorial

The Transactions on Computational Science journal is part of the Springer series *Lecture Notes in Computer Science*, and is devoted to the gamut of computational science issues, from theoretical aspects to application-dependent studies and the validation of emerging technologies.

The journal focuses on original high-quality research in the realm of computational science in parallel and distributed environments, encompassing the facilitating theoretical foundations and the applications of large-scale computations and massive data processing. Practitioners and researchers share computational techniques and solutions in the area, identify new issues, and shape future directions for research, also enabling industrial users to apply the techniques presented.

The current issue consists of two parts: Part I is devoted to neural and social networks, and Part II to geometric modeling and simulation. Part I is comprised of four papers, spanning areas of information–driven on-line social networks, neural networks, collaborative memories, and stability controls in multi-agent networked environments. Part II is comprised of four papers united by the theme of geometric modeling. These papers cover the topics of shape reconstruction from planar contours, sharp feature preservation through wavelets, protein structure determination based on the beta-complex, and fast empty volume computation in molecular systems. The first article of the current issue was invited to the journal as one of the top papers from the ACM CyberWorlds 2012 conference, while all other articles were submitted as regular papers.

We would like to extend our sincere appreciation to the TCS Editorial Board and external reviewers for their dedication and insights in preparing this issue. We would also like to thank all of the authors for submitting their papers to the journal. We would like to express our gratitude to the LNCS editorial staff of Springer, who supported us at every stage of the project.

We our hope that this issue will be a valuable resource for Transactions on Computational Science readers and will stimulate further research into the vibrant area of computational science applications.

December 2013

Marina L. Gavrilova
C.J. Kenneth Tan

LNCS Transactions on Computational Science – Editorial Board

Table of Contents

Part I

Neural and Social Networks

Weibo: An Information-Driven Online Social Network[*]

Zhengbiao Guo, Zhitang Li, Hao Tu, and Da Xie

Computer Science Department, Huazhong University of Science & Technology, WH, China
{zhengbiaoguo,xiedaa}@gmail.com, {leeying,tuhao}@hust.edu.cn

Abstract. Online social network (OSN) is becoming more and more prevalent currently. Some literature about it has been published, but few papers talked about Sina Weibo, which is the largest microblog in China. Weibo increases rapidly and draws 100 million users within a year and a half, and users of Weibo are Chinese, who enjoy a different culture.

We crawled Weibo for one month and collected 1.12 million user profiles. Using this dataset, we study the dynamics and the characteristic path length of the network, some core users and the reciprocal rate. Based on our results, we show the topological characteristics of Sina Weibo and deduce what people use Sina Weibo for. We believe our findings can be used to understand current large-scale OSN for future research on tweet propagation and hot topic prediction, and to provide useful and practical hints for future design of large-scale OSN system.

Keywords: measurement, structure, Weibo, Sina microblog, online social network.

1 Introduction

OSN is more and more popular throughout the world, and it is expected to be the next-generation communication system, for these systems make users communicate with each other conveniently. Many important things are propagated through the OSNs, for example, Egyptian revolution [1], Iran's Protests [2] and people look for their lost child using Sina Weibo in China [3]. In fact, you can know the events happened in China everyday using Weibo.

Some papers talked about Twitter [4] but few papers did research on Weibo [5], which is the top microblogging service in China. Sina Weibo launched on Aug 28, 2009 and attracted 100 million users within a year and a half. In Weibo, users can send and read text-based posts composed of up to 140 characters, called tweets, which are displayed on the user's profile page. Users can subscribe to other users' tweets – this is known as followings and subscribers are known as followers. Unlike Facebook, a user can follow any other user, and the user being followed need not follow back, which makes microblog a different type of OSN.

[*] Supported by the FRFCU grant 2011QN045, NSF of Hubei Province grant 2010CDB02306 and CNGI grant CNGI-122.

There are three reasons made us do research on Sina Weibo: (1) Weibo grows two times faster than Twitter; (2) users of Weibo enjoy a different culture; (3) users of Twitter only can use less than 140 characters to express themselves but Weibo can use 140 Chinese characters, which contain more information. In addition, there are many differences between Twitter and Weibo in detail, for example, tweets in Weibo could contain images and videos besides text message and links, and dealing with every tweet's reply and comments is different between Weibo and Twitter.

In our opinion, there are three basic questions in microblog system: (i) How does people connected with each other and form the network, (ii) What are the users talking about in Weibo, and (ii) How does the information transmit through the network. The goal of this work is only to study the topological characteristics of Weibo and infer what people use Weibo for.

Our work is a practical measurement using data collected from Weibo. We began to crawl Weibo in Oct, 2010 after Weibo opened its API. We take one month's dataset to analyze the overlay of Weibo, and the dataset includes 1.12 million users, 2% of the whole users of Weibo at that time. We studied the distributions of followings and followers, the relation between followers, followings and tweets, and the dynamics of users' followers. Then, we analyzed the characteristic path length of Weibo, and the reciprocal rate of users. Finally, we analyzed the topological characteristics of the network formed by verified users (users who are verified by Weibo). Based on our results, we classify OSNs into two types – information-driven and relationship-driven.

Structure determines function. Discovering the characteristics of Weibo's structure can make us know such systems clearly, and it is also the basic research before we do other research on OSN, for example, tweets propagation and hot topic prediction. Our work is the first quantitative study on the topological characteristics of Weibo, and makes insight into its structure. According to the results, we also deduce what people use Weibo for, which is useful for future design of large-scale OSN system. In addition, OSN is the map of the real society. Knowing the structure of Weibo is helpful for us to understand the Chinese society.

The rest of this paper is organized as follows: We cover the related work in Section 2, and show the overview of Weibo in Section 3. We analyze the topological characteristics of Weibo in Section 4 and discuss our results in Section 5. We conclude in Section 6.

2 Related Work

There are three basic research areas in microblog system, and we talk about the related work in such areas in this section.

2.1 The Structure of OSN

Mislove et al. [11] measured and analyzed the structural properties of Orkut, Youtube, Flickr and LiveJournal. They observed the indegree of user nodes tended to match the outdegree which was different from our results. In addition, they talked about groups and cores of these OSNs. Kwak et al. [7] basically analyzed the structure of Twitter, which was a little different from Weibo and used different methods to identify users' influence on Twitter. Wu et al. [10] used Twitter lists to classify the users in Twitter into different categories and found 50% of URLs were generated by 20K elite users. This paper also showed users in the same categories show some level of homophily. Java et al. [12] studied the topological and geographical properties of Twitter's social network. They found users talk about their daily activities and to seek or share information using Twitter, which is similar as the results in our paper; they also classified users using CPM. Huberman et al. [9] reported that the number of friends was actually smaller than the number of followers or followings, which is the same as our result.

2.2 The Content of Tweets

J. Weng et al. [14] used an extension of PageRank algorithm to measure the influence of users in Twitter, and used LDA to identify latent topic information. M. Cataldi et al. [15] gave a topic detection technique that permits to retrieve in real-time the most emergent topics expressed by the community, using a novel aging theory and PageRank. P.Owen et al. [16] used lucene to analyze the content of Twitter, and TF-IDF to recommend real-time topical news. L. Yu et al. [17] analyzed Weibo, and showed the trends in Weibo are created almost entirely due to retweets of media content such as jokes, images and videos.

2.3 The Diffusion of Tweets

Kwak et al. [7] analyzed the tweets of trending topics, talked about the tweets' diffusion and observed retweets reach a large audience and spread quickly. Wu et al. [10] talked about the two-steps diffusion in Twitter, investigated the flow of information among different categories, finding that many of the tweets generated by core users reaches the masses indirectly via a large population of intermediaries, and different content types exhibited dramatically different characteristic lifespans. David et al. [13] talked about a single piece of information spreads on a global scale using Internet chain-letter data and find that the progress of these chain letters precedes in a narrow but very deep tree-like pattern, continuing for several hundred steps.

Our work focuses on the characteristics of Weibo's structure which is not studied before, and infers what people use Sina Weibo for. We analyze four characteristics of Weibo, and using these factors, we classify OSN into two different types.

3 Overview of Weibo

Weibo is a self-organization communication application. With more and more people using microblogging service, the scale of Weibo becomes larger and larger. There are three major parts of Weibo: users, overlay and tweets. Users connect with each other by self-organization and form the overlay, and then they share information through the overlay.

When a new user joins Weibo, he should register Weibo and get a unique ID, and then he takes some existing users as his followings and waits for others to follow him. When the new user joins the network, he begins to use the overlay to receive or post tweets.

From the join process of Weibo, we know the way users choose their followings decides the topology of the system, and we talk about it as follows. There are about three ways one user chooses his followings: (1) he follows the users he knows; (2) he follows the users who are followed by the users he follows now; (3) he searches the users who have the same tags or interests with him. At the same time, he can cancel his followings that he doesn't like. We can infer the overlay of Weibo will change frequently because new users join in and old users change their followings. Thus, we can build a model to show how Weibo's overlay evolves with the time.

- Startup: there are M users and e links in the initial network;
- Growth: every new user joins in the network with m followings(m<2000);
- Link: the new user chooses m old users from Weibo and follows them;
- Evolution: old users change their followings with time;

In this paper, we analyze the characteristics of Weibo, and we also want to find the way how to fix the parameters in our evolution model.

4 Analysis of Network Structure

In this section, we characterize the structural properties of Weibo, such as degree distribution, the characteristic path length, and reciprocal rate.

4.1 Dataset

Weibo offers API [6] in July, 2010, and we use it to collect data. Our first version of crawler began to work in Oct, 2010. In this paper, the dataset was collected from Nov 1, 2010 to Dec 1, 2010. The overlay of Weibo may change, so we only take one month's data to analyze the topology of Weibo. We collected profiles of users and their tweets, and the profiles include full name, location, gender, number of following, number of follower, number of tweets, list of following and whether it is verified.

Weibo rate-limits 1,000 requests per hour per every user, and rate-limits 10,000 requests per hour per white-listed IP, which make it hard to crawl the overlay rapidly. The dataset includes 1.12 million users' information, which counts for 2% of the total users of Weibo, and Weibo had about 55 million users at the moment. Additionally, users of Weibo can be classified into two kinds: verified user (Vuser) and common user (Cuser). Cusers are users who aren't verified by Weibo.

4.2 Followings, Followers vs. Tweets

Number of followings, followers and tweets are three basic factors can be used to analyze the structure of Weibo. As we introduced in section 4.1, there are two types of user in Weibo: Vuser and Cuser, which play different roles in Weibo, and we analyze them separately. Our dataset has 1,125,044 users, which includes 1,089,676 Cusers and 35,368 Vusers. Fig.1 displays the distribution of Cuser's number of followings and followers. Fig.1(a) shows the complementary cumulative distribution function (CCDF) of Cuser's number of followings; Fig.1(b) shows the log-log graph of Cuser's number of followings; Fig.1(c) shows the CCDF of Cuser's number of followers and Fig.1(d) shows the log-log graph of Cuser's number of followers.

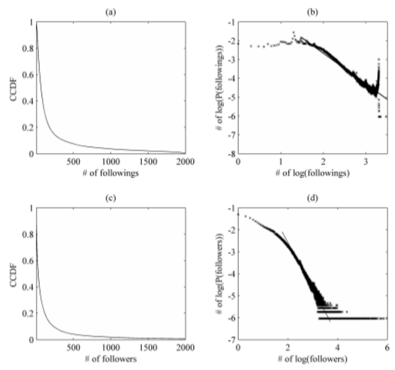


Fig. 1. # of followings & followers of Cusers

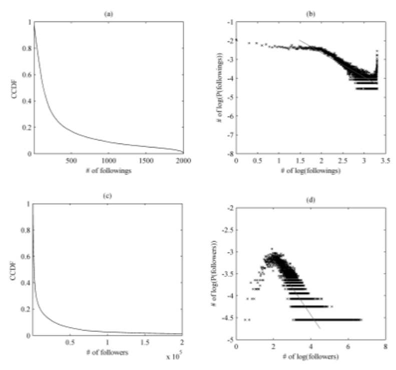


Fig. 2. # of followings & followers of Vusers

Cusers have 148 followers, 167 followings and 292 tweets on average. The ratio of the followings' number less than 30 is 20%, less than 500 is 92.5% and more than 500 is 7.5%. We first explain the glitches in Fig.1(b). The first occurs at x=1.4 which indicates the followings' number is 30. Weibo gives new users an initial set of 30 people to follow and most of the new users follow them. The second occurs at x=3.3 which means the followings' number is 2000. Users can only follow less than 2000 people in Weibo but sometimes the system may make mistakes, so this glitch happens. The line in Fig.1(b) from 30 to 2000 fits to a power-law distribution with the exponent of 1.63.

The ratio of the followers' number less than 54 is 54%, less than 232 is 90% and less than 1,515 is 99%. The line in Fig.1(d) from 54 to 9,000 fits to a power-law distribution with the exponent of 2.1. The ratio of Cusers who have more than 10,000 followers is less than 0.11% which makes the end of the line in Fig.1(d) flat.

The dataset has 35,368 Vusers, and Vusers have 17,716 followers, 307 followings and 649 tweets on average. The ratio of the followings' number less than 30 is 15%, less than 1000 is 91.2%. The line in Fig.2(b) from 30 to 2000 fits to a power-law distribution with the exponent of 1.29.

The ratio of the followers' number less than 100 is 5.16%; less than 10,000 is 79.9%. The line in Fig.2(d) from 100 to 10,000 fits to a power-law distribution with the exponent of 2.63.

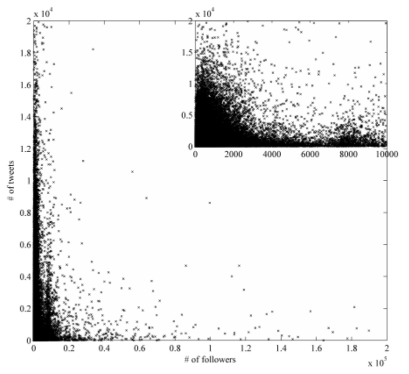


Fig. 3. # of followers & tweets per users

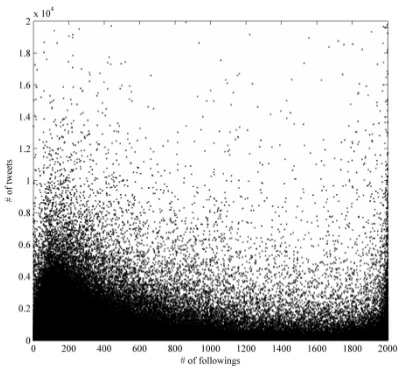


Fig. 4. # of followings & tweets per users

We also analyze the correlation between the number of followers and that of tweets. In Twitter, users have more followers send more tweets [7]. In Fig.3, It's clear the number of tweets is not increasing with the increasing number of followers. Users with more than 10,000 followers didn't send many tweets, and the number of tweets is less than 2,000 on average. We also show the distribution from 0 to 10,000 in detail, and we can see that some users with few followers send more tweets. Most of the users who have more followers are Vusers and they are recommended by the system, but they send a few tweets. The users who have few followers are Cusers and they send many tweets. Fig.4 shows the correlation between the number of followings and that of tweets. We can infer that the number of tweets is also not increasing with the increasing number of followings.

From Fig.3 and Fig.4, we can say the number of tweets is not related to the number of user's follower and following. We also know though one user send many tweets, he may not attract many followers.

4.3 Change of User's Follower

We notice the overlay of Weibo is dynamic and we collect eleven days' data (from 2011-01-05 to 2011-01-16) which includes 100 Cusers' follower numbers every day.

The number of users' follower changes frequently and a user gain 23,495 new followers in one day. The average increment of the users' followers in our dataset is 2,932. In Facebook, one has about 130 friends [8], but in Weibo the number is 872 and the links between users change frequently as shown in Fig.5. We can infer most of the user's followers are not his friends, and they didn't know each other in the real society – a conclusion that is reported by paper [9].

We also compare the number of users' follower and the number of users' increased follower. They are correlative and the correlation coefficient is 0.55. That's to say when you have more followers now, and you will get more followers in the future, which shows the Matthew effect is significant in Weibo. We didn't collect Vusers' change of follower for we didn't get such API, and it is our future work.

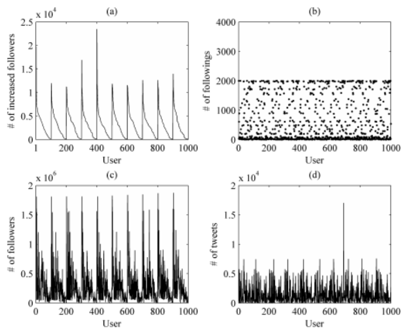


Fig. 5. The change of user's follower

So we can say that the user coming to Weibo is not for friendship and this makes it different from Facebook.

4.4 Characteristic Path Length of Weibo

The Characteristic path length (CPL) is another basic property, which should be discussed to understand the overlay of Weibo. This characteristic makes us know how quickly the tweets delivered from the origin user to others in Weibo, and helps us to study the information diffusion on OSN.

The link between two users in Weibo is directed. We can't crawl all the relationships in Weibo in a short time, so we didn't measure the CPL of Weibo directly. Although we can get all the relationships, we can't tell the true CPL of Weibo for the dynamic of the overlay as shown in section 4.3. We built an analytical model to evaluate it as follow.

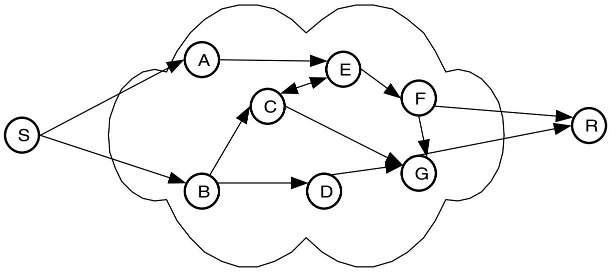


Fig. 6. Tweet's transmission

To evaluate the CPL of Weibo, we show the paths one tweet sent from S to R as shown in Fig.6, and the shortest length from S to R is the distance between them. We want to find out how many steps should take from S to R at least, and then we change the shape of Fig.6 into Fig.7.

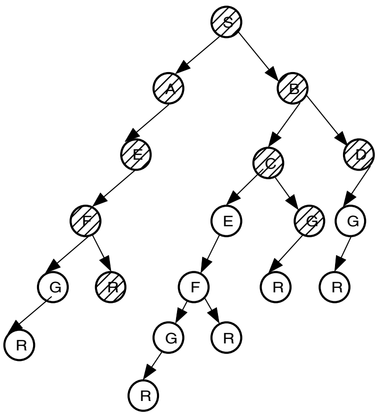


Fig. 7. The deformation of Fig 6

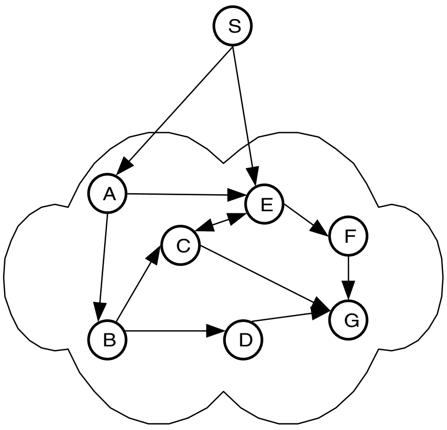


Fig. 8. Tweet's diffusion

Fig.7 is generated from Fig.6 using Breath-First Search (BFS), dark notes mean they appeared for the first time using BFS and white nodes mean they appeared before. We define a function δ (n) as

$$\delta(\mathrm{n}) = \begin{cases} 1, & if\ \pi_n \neq \pi_{n-1} \\ 0, & else \end{cases} \tag{1}$$

δ (n) is 1 when the user n first appeared during the BFS, and π_n includes all the non-repeating users when we searched the No.n users. As we assume the links between users are random. For an overlay of size N, we have

$$\mathrm{P}[\delta(\mathrm{n}) = 1] = \frac{N - f(\pi_{n-1})}{N} \tag{2}$$

Where $f(\pi_n)$ stands for the number of π_n. So $\delta(\mathrm{n}) = f(\pi_n) - f(\pi_{n-1})$, taking expectation on both sides of Eq. (2), we have

$$E\big(f(\pi_n) - f(\pi_{n-1})\big) = E\big(\delta(\mathrm{n})\big) = \frac{N - E(f(\pi_{n-1}))}{N} \tag{3}$$

Which follows that

$$E\big(f(\pi_n)\big) = 1 + \frac{N-1}{N} E\big(f(\pi_{n-1})\big) \tag{4}$$

Since $f(\pi_1) = 1$, iteratively solving Eq.(4) leads to

$$E\big(f(\pi_n)\big) = N\left(1 - \left(\frac{N-1}{N}\right)^n\right) > N\left(1 - e^{-\frac{n}{N}}\right) \tag{5}$$

Let n_k denotes the last user whose distance from S is k, and the average distance from S to all other users is as follow

$$\mathrm{d} = \frac{1}{N} \sum_{k=1}^{\infty} k * E(f(\pi_{n_k}) - f(\pi_{n_{k-1}})) \tag{6}$$

Since $\lim_{k\to\infty}\frac{E(f(\pi_{n_k}))}{N}=1$, thus we have

$$\begin{aligned} d &= \sum_{k=1}^{\infty} k*((1-\frac{E(f(\pi_{n_{k-1}}))}{N})-(1-\frac{E(f(\pi_{n_k}))}{N})) \\ &= \sum_{k=0}^{\infty}(1-\frac{E(f(\pi_{n_k}))}{N}) \\ &< \sum_{k=0}^{\infty} e^{-\frac{n_k}{N}} \end{aligned} \tag{7}$$

We assume every user in Weibo has the same number of followers and followings and they have 148 followers and 167 followings on average as shown in section 4.2. Let L denote the users' degree which is bigger than 100, and except S the other users have L-1 children. It follows that

$$n_k = \frac{L*(L-1)^k-2}{L-2} \tag{8}$$

We divide Eq. (7) into two parts: the first part is from $k=0$ to $k=\log_{L-1}N$, and the second from $k=1+\log_{L-1}N$ to infinity, that is,

$$\begin{aligned} d &< \sum_{k=0}^{\log_{L-1}N} e^{-\frac{L*(L-1)^k-2}{N(L-2)}} + \sum_{k=\log_{L-1}N+1}^{\infty} e^{-\frac{L*(L-1)^k-2}{N(L-2)}} \\ &< \log_{L-1}N+1+\sum_{m=0}^{\infty} e^{-\frac{LN(L-1)^{m+1}-2}{(L-2)N}} \\ &< \log_{L-1}N+1+\sum_{m=0}^{\infty} e^{-(L-1)^m} \end{aligned} \tag{9}$$

If $L\geq 3$, we have $(L-1)^m \geq (L-1)m$ and hence $e^{-(L-1)^m} \leq e^{-(L-1)m}$, which follows that,

$$\begin{aligned} d &< \log_{L-1}N+1+\sum_{m=0}^{\infty} e^{-(L-1)^m} \\ &< \log_{L-1}N+1+\sum_{m=0}^{\infty} e^{-(L-1)m} \\ &= \log_{L-1}N+1+\frac{1}{(1-e^{-(L-1)})} \\ &< \log_{L-1}N+3 \end{aligned} \tag{10}$$

From Eq. (10) we know that the average length from S to all other users in Weibo is bounded by $O(\log_{L-1}N)$.

Now we talk about how many hops should be needed if S wants to send a tweet to R. There are 55 million users when we crawl Weibo, and every user has 148 followers and 167 followings on average. When S wants to send a tweet to R, he will ask his followers and followings to help him, and others will also ask their followers and followings to help them. So L will be 315, and the hops will be 6.09 on average.

In fact, no user demands his followers and followings to help him to send his tweets, so the tweets will be diffused randomly as shown in Fig.8. In this condition, the L will be the number of users' follower and the hops will be 6.56. Here we only use the number of Cuser, and the Vusers' follower and following number are bigger which make the hops less than 6.

In Weibo, tracing the path of one tweet is hard and no API can be used. From our results, we know the tweet will flow the whole overlay using a few hops. This indicates when one tweet is published, it will be known by all the others in a short time, as shown in Eq. (10). Facebook also has a short CPL for user with 130 friends on average [8]. However, when one message born in Facebook, it may diffuse slowly because users only want to share their information with their friends and this stops the message from widely transmitting, which makes Facebook unsuitable for the timely dissemination of information.

4.5 Vuser's Overlay

In our opinion, Vuser may play an important role in Weibo, and we analyze their overlay particularly. Our dataset has 35,368 Vusers, and Vusers have 17,716 followers, 307 followings and 649 tweets on average, which is more than Cusers'. Next we will discuss the links' characteristic of Vuser.

We count all the numbers of Vusers' followers and it is up to 695,968,649. Users in Weibo follow 98 users [5], and there are about 55 million users at that time. The sum of following links is about 5.39 billion, and the total number of Vusers' follower accounts for 12.91% of all the following links. The total number of Vusers' following is 10,857,976 and 2,054,311 of them link to Vusers, which take about 18.92% of all the following links. Weibo classifies Vusers into many groups according to users' career.

Table 1. Basic statistics for a number of groups

Group	N	M	D	<k>	C
Movie Star	1200	21457	9	35	0.16
TV Anchor	1200	18172	9	30	0.25
Radio DJ	1200	16994	8	28	0.31
Singer	801	12028	8	30	0.20
Journalist	849	11335	7	26	0.21
Police	292	11303	5	77	0.42
IT	711	9048	7	25	0.19
IT Executive	223	4133	7	37	0.29

In fact, we didn't collect all the Vusers and some of the Vuser don't draw many followers. From our results, there is a core in Weibo and users from the core attract most of the users in Weibo and they connected with each other strongly. Finding the true core of Weibo is our future work.

The properties measured are: group, directed or undirected; total number of vertices N; total number of edges M; diameter of the group D; mean degree <k>; and clustering coefficient C.

From Table 1, we can see the diameters of the groups are short though the graphs are directed and they are less than 8 generally. The clustering coefficients of the groups are bigger than O (1/N) and less than 1, which means the groups have Cluster Effect and users in some groups connected with each other strongly.

In summary, there is a densely-connected core in Weibo and some Vusers have the characters as core users. They connect with each other strongly and attracted many other users in Weibo. There are also many groups among core users and the diameters of these groups are short, which make the tweets transmit quickly. In this section, we just analyze the Vusers as core users and the groups classified according to Vusers' career.

The existing of core users makes Weibo different from other OSNs, because few people can be familiars with most of the users in an OSN. The result also indicates users in Weibo aren't for friendships but for information.

4.6 Reciprocal Rate

We notice most of the users don't follow each other, that's to say most of the links between users are one-way, which is a significant characteristic of Weibo. We take the number of user's followers as $\mathrm{F_w}$, the number of user's followings as $\mathrm{F_g}$, the number of users existing in one's followers and followings as X and the reciprocal rate as $\mathrm{R_r}$. Thus, the reciprocal rate of one user is defined as Eq.(11).

$$R_r = \frac{X}{(F_w - X) + (F_g - X) + X} = \frac{X}{F_w + F_g - X} \tag{11}$$

As shown in section 4.2, the figure of Vusers' followers is big and it's hard to get the X of Vuser. We take Eq.(12) to estimate Vuser's reciprocal rate.

$$R_r = \frac{X}{(\mathrm{F_w} - X) + (\mathrm{F_g} - \mathrm{X}) + X} < \frac{\mathrm{F_g}}{\mathrm{F_w} + \mathrm{F_g} - \mathrm{F_g}} = \frac{\mathrm{F_g}}{\mathrm{F_w}} \tag{12}$$

And

$$\overline{R_r} = \frac{1}{N} \sum_{i=1}^{N} R_{ri} \tag{13}$$

The reciprocal rate of Vuser is less than 0.7% and of Cusers is about 16.9%. In Twitter, this result is about 22.1% [7]. In Facebook, users follow each other and the reciprocal rate will be 1. Thus, OSN can be classified into two types: one type is users keep in touch with each other and they may look for friends, and the other is most users didn't follow each other and they may look for information.

5 Discussion

Our measurement presents many interesting phenomena, and we want to discuss implications of them in this section.

Our measurements indicate different users have different numbers of followers, followings, tweets; the reciprocal rate is low; and the overlay is dynamic; the characteristic path length of Weibo is short. From these results, we want to classify online social networks into two types as shown in Fig.9.

OSNs only contain two basic factors: users and tweets, and users of such systems could find friends or seek tweets. Different OSNs provide different functions, thus, we distinguish them apart by using four rules: (1) the dynamics of user's following and follower; (2) the reciprocal rate of users; (3) the users' follower number; (4) the information's transmission speed. If one is a relationship-driven OSN, its overlay changes slowly, and users follow each other because they are familiars and they want to keep in touch with each other which make users' followings number small. No people can make friends with most of the other users, so the core of such networks is insignificant. Users in a relationship-driven OSN will show some private things among them and don't want to share these things to all the people in the network which will prevent the dissemination of the information. Information-driven OSN is opposite to relationship-driven OSN and user coming here seeks for information and doesn't care whether he is familiar with his followings or followers.

We think the future OSN systems will look like Google+, users in which can make different users into different circles and share different information with them. Such systems are hybrid platforms and one system provides both need of users, including friendship and information. However, the basic types of OSN system are information-driven OSN and friendship-driven OSN.

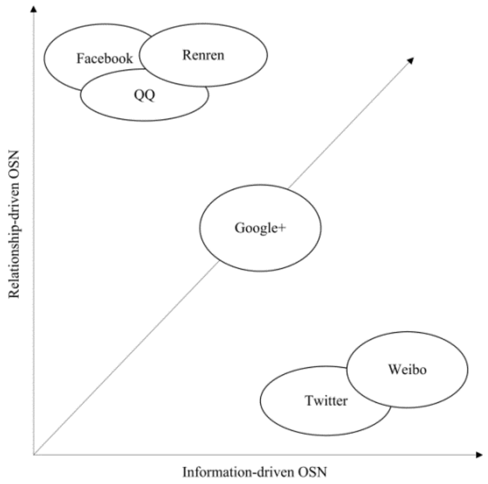


Fig. 9. Two basic types of online social network

In addition, the real social network is formed by people's relationship. Online social network gives a chance to make people find more friends. When the online links become the true relationships in the real social network, the real social world will change, so OSNs will impact the physical world in the future.

6 Conclusion

Three months after Weibo opened its API, we started to crawl and collected one month's data including 1.12 million user profiles to analyze its overlay. We analyzed the relationship between users' following, follower and tweets, and found the number of tweets would not increase with the number of following or follower. The following and follower distribution are fit for power-law in some interval. We also paid attention to the dynamics of Weibo and found that links between users change frequently, which makes Weibo different from other OSNs and social networks. We proved that the characteristic path length of Weibo is short and tweets can flow from one user to another in less than 6 hops, which is similar to Twitter's [7]. We also discovered that there is a core in Weibo which attract about 12.91% of all the users' following links and users in the core connect with each other strongly. Then, we defined how to count the reciprocal rate in Weibo and show the reciprocal rate of Weibo is less than 16.9%, which is lower than Twitter's. From these results we deduct that Weibo is different from human social network and other OSN. Thus, we classify OSN into two kinds: information-driven OSN and relationship-driven OSN.

Much work is still to be done. We should find the true core of Weibo and the true groups of the core network because Weibo classified the Vusers into different groups according to their career and many users who weren't verified also attracted too many users. Understanding the overlay of Weibo can help us to determine this type of OSN clearly, which may change human social network in the future.

Acknowledgements. We are grateful to Jin Ding, Jiqiao Cheng and Chunyong Shi for their helpful work. We also thank anonymous reviewers for their valuable comments. This work was supported by NSF of Hubei Province [2010CDB02306], CNGI [CNGI-122] and the Fundamental Research Funds for the Central Universities [2011QN045].

References

1. `http://en.wikipedia.org/wiki/2011_Egyptian_revolution` (updata: April 03, 2011)
2. `http://www.time.com/time/world/article/0,8599,1905125,00.html` (updata: April 03, 2011)
3. `http://online.wsj.com/article/SB10001424052748703716904576133912072642724.html` (updata: April 03, 2011)
4. `http://www.twitter.com`
5. `http://www.weibo.com`

6. http://open.t.sina.com.cn
7. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a Social Network or a News Media? In: WWW 2010, Raleigh, North Carolina, USA, April 26–30 (2010)
8. http://www.facebook.com/press/info.php?statistics (updata: May 03, 2011)
9. Huberman, B.A., Romero, D.M., Wu, F.: Social networks that matter: Twitter under the microscope. arXiv:0812.1045v1 (December 2008)
10. Wu, S., Hofman, J., Mason, W., Watts, D.: Who Says What to Whom on Twitter. In: World Wide Web Conference Series - WWW, pp. 705–714 (2011)
11. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: IMC 2007, San Diego, California, USA, October 24-26 (2007)
12. Java, A., Song, X., Finin, T., Tseng, B.: Why we twitter: understanding microblogging usage and communities. In: Proc. of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis. ACM (2007)
13. Liben-Nowell, D., Kleinberg, J.: Tracing information flow on a global scale using Internet chain-letter data. Proc. of the National Academy of Sciences 105(12), 4633–4638 (2008)
14. Weng, J., Lim, E.P., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential twitterers. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 261–270. ACM (2010)
15. Cataldi, M., Di Caro, L., Schifanella, C.: Emerging topic detection on Twitter based on temporal and social terms evaluation. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining. ACM, Washington, D.C (2010)
16. Owen, P., Kevin, M., Barry, S.: Using Twitter to recommend real-time topical news. In: RecSys 2009, New York, USA, October 23–25 (2009)
17. Yu, L., Asur, S., Huberman, B.: What Trends in Chinese Social Media. In: The 5th SNA-KDD Workshop 2011 (SNA-KDD 2011), SanDiego CA USA, August 21 (2011)

Collaborative Memories in Clusters: Opportunities and Challenges

Ahmad Samih[1], Ren Wang[2], Christian Maciocco[2], Mazen Kharbutli[3], and Yan Solihin[4]

[1] Intel Architecture Group, Austin, Texas, USA
ahmad.a.samih@intel.com
http://www4.ncsu.edu/~aasamih
[2] Intel Labs, Hillsboro, Oregon, USA
{ren.wang,christian.maciocco}@intel.com
[3] Jordan University of Science and Technology, Irbid, Jordan
kharbutli@just.edu.jo
[4] North Carolina State University, Raleigh, NC, USA
solihin@ncsu.edu

Abstract. Highly-integrated distributed systems such as Intel Micro Server and SeaMicro Server are increasingly becoming a popular server architecture. Designers of such systems face interesting memory hierarchy design challenges while attempting to reduce/eliminate the notorious disk storage swapping. Disk swapping activities slow down applications' execution drastically. Swapping to the free remote memory - near by nodes, through *Memory Collaboration* has demonstrated its cost-effectiveness compared to overprovisioning memory for peak load requirements. Recent studies propose several ways to access the under-utilized remote memory in static system configurations, without detailed exploration of dynamic memory collaboration. Dynamic collaboration is an important aspect given the run-time memory usage fluctuations in clustered systems. Furthermore, with the growing interest in memory collaboration, it is crucial to understand the existing performance bottlenecks, overheads, and potential optimizations.

In this paper we address these two issues. First, we propose an Autonomous Collaborative Memory System (ACMS) that manages memory resources dynamically at run time, to optimize performance, and provide QoS measures for nodes engaging in the system. We implement a prototype realizing the proposed ACMS, experiment with a wide range of real-world applications, and show up to 3x performance speedup compared to a non-collaborative memory system, without perceivable performance impact on nodes that provide memory. Second, we analyze, in depth, the end-to-end memory collaboration overhead and bottlenecks. Based on this analysis, we provide insights on several corresponding optimizations to further improve the performance.

1 Introduction

With every new software generation, applications' memory footprints are growing exponentially in two dimensions–horizontally due to an increase in their data set, and vertically due to additional software layers. This fast memory requirement growth outpaces the growth in the capacity of current memory modules (RAMs) [18]. This leads the OS's virtual memory manager to resort to swapping to storage devices, e.g., Hard Disk Drives (HDDs) and Solid State Drives (SSDs). Swapping devices such as HDDs, or even SSDs operate at several orders of magnitude slower compared to main memory modules [29]. Excessive paging activity to and from the swapping device renders a system crawling as the CPU is mostly waiting for I/O activity. The performance degradation, in turn, poses serious power implications since the slow execution keeps the CPU and system in high power state longer than necessary.

Recently, we have seen the trend of the fast development of high density, low power, and highly integrated distributed systems such as clustered systems (e.g., Seamicro's SM1000-64HD[32] and Intel's microServer [8]). In these systems, hundreds or even thousands of independent computing nodes are encapsulated within a single platform. This, therefore, poses interesting challenges as to how designers could restructure the memory hierarchy to achieve optimal performance given a certain peak load requirement, with consideration of cost and energy budgets.

There is a spectrum of solutions that attempt to bridge the vast performance gap between the local memory and the disk storage in clustered systems by avoiding swapping activity as much as possible. One end of the spectrum suggests over provisioning the system with more precious resources. Over provisioning may range from installing more physical memory, adding dedicated memcached servers [1], leveraging a hybrid memory system of PCM, PRAM, and DRAM [29,31,9], or even adding a dedicated storage server that stores all data on their main memory (RAMs); namely the RAMCloud [25]. While over provisioning the system to accommodate all memory needs solves the problem, it comes with prohibitive costs and excessive power budget.

The other end of the spectrum suggests a more cost-effective design by improving the aggregate cluster memory utilization. At a high level, improving cluster utilization involves making use of idle memory located at remote nodes, namely Memory Collaboration. Memory Collaboration can be categorized into two approaches:*remote memory swapping* [17,24,19,20,43], and *remote memory mapping* [22,43].

Remote memory mapping techniques consider the remote memory as an extension to the local memory space. Such techniques usually require inflexible malloc-like APIs to manage local and remote memory resources, or recompilation of applications to distribute the statically defined memory structures (i.e., arrays) onto local and remote memories. Further, some remote memory mapping techniques such as [22] requires user intervention to explicitly define the availability of memory space at remote nodes.

In this paper, we focus on the other approach, remote memory swapping, which considers remote memory as a swap device. Approaches that fall into this category have demonstrated the ability to be deployed transparently with little/no modification to the OS or the running applications, while at the same time partially filling the performance gap between local memory and hard disk with a cost-effective design. However, these proposals often focus on static system configurations and lack the detailed investigation, implementation and evaluation on the aspect of dynamically detecting, provisioning, and utilizing remote memory to optimize performance for the whole cluster. Furthermore, as remote memory swapping becomes an appealing approach, it is critically important to understand the performance bottlenecks that exist in current systems and how such bottlenecks could be potentially removed or mitigated.

In this paper, we address these two concerns and make the following major contributions:

1. We propose a system architecture and a memory acquisition protocol to perform robust, efficient, and autonomous memory collaboration across multiple nodes within a cluster. The proposed protocol allows for multiple nodes to dynamically discover, allocate, and deallocate remote memory based on local memory requirements. We demonstrate the feasibility and benefit of the proposed ACMS by developing a prototype and evaluating real-world applications. Our prototype results show that an ACMS-based system can *adapt* to workload dynamics and memory variations, and achieves up to 3x speedup compared to a non-collaborative memory system.
2. We pinpoint and analyze the major performance bottlenecks and overheads during the lifetime of the remote memory swapping approach. Our investigation shows that the network stack and Linux kernel swapping process are the major bottlenecks.
3. We further study several optimizations to improve remote memory swapping performance. Based on our investigation, we give insights into how to take advantage of software/hardware optimizations to significantly speed up the remote memory access.

The rest of the paper is organized as following. Section 2 motivates our dynamic approach for collaborative memories in clustered architectures. Section 3 reviews the related work. Section 4 describes the design of the proposed Autonomous Collaborative Memory System (ACMS). Section 5 describes the implementation details and prototyping of ACMS. Section 6 describes the evaluation methodology and provides the results from our evaluations and analyzes the findings. Section 7 provides insightful analysis into the remote swapping overhead. Section 8 concludes the work and discusses the future works.

2 Motivation for Dynamic Memory Collaboration

In Section 1, we have discussed that limited memory resources force the system to resort to slow storage devices which has major implications on performance and power dissipation. For single-node systems, if over provisioning is not an option, the OS has to start paging to and from the disk and therefore suffer the high latencies.

With multi-node clusters [32,8], the overall picture is different. Some nodes in the cluster may over utilize their memory system, while other nodes may under-utilize them, and the dynamics often change over time. This imbalance in memory usage across different nodes in a cluster has motivated our work to investigate techniques to make use of the under-utilized, and fast remote memory, over using the slow storage device.

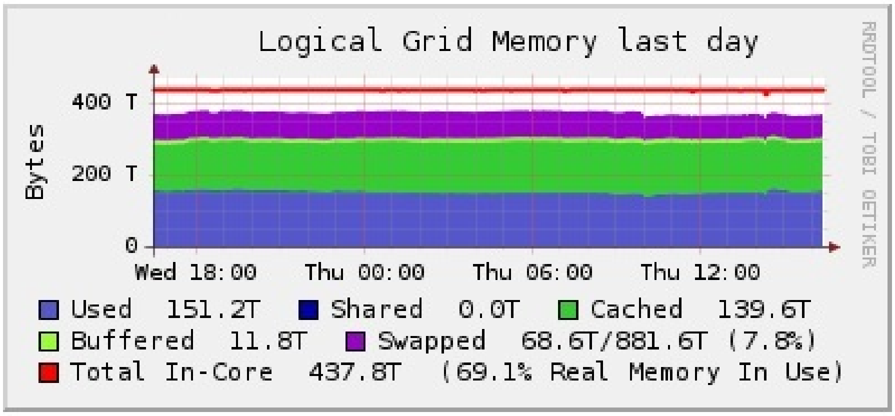


Fig. 1. Example of a Data Center Memory Usage Imbalance

Figure 1 shows the memory usage during a typical workday in a typical data center cluster. The data is collected using Ganglia tool [21], which is a scalable monitoring system tool for high-performance computing systems such as clusters and grids. As can be seen in the figure, the aggregate memory in the cluster reaches 437TB. However, only 69% of this aggregate memory is being utilized (i.e., used + cached + buffered / Total). Despite the fact that the aggregate utilization is far from being 100%, there is about 68TB of data residing in swap devices. This demonstrates that some nodes are over utilizing their memories, while others have free memory available potentially for donation. Since memory nodes are physically private to each node, this free memory will not be utilized by default by other nodes in the cluster. *This motivates the need to have a collaborative memory framework to manage aggregate memory resources within the cluster in order to reduce storage device swapping activity and improve performance.*

Furthermore, studies have shown that local memory requirements can vary drastically [18] over time based on multiple factors such as workload variations, orthogonal execution phases, etc. Moreover, thread migration from one node to another (e.g., VMware's vMotion technology [42]), shifts the memory demand from the source node to the destination. Managing the drastic spatial/temporal variation of memory requirement in multi-node clusters is no easy task. It calls for a stable, run-time mechanism to *classify* and continually *reclassify* nodes based on their memory requirements, and to dynamically assign remote memory to achieve optimized performance and energy-efficient memory collaboration.

To address these concerns, we propose a *fully* autonomous memory acquisition protocol that can be implemented on various interconnects. The protocol facilitates memory detection, memory exchange, and memory reclaim across nodes within the cluster dynamically at run time. The dynamic control protocol for memory collaboration is, to the best of our knowledge, novel.

3 Related Work

There is a rich body of work that has studied the problem of managing capacity at different levels of the memory hierarchy [33,34,5,7,27,44,4,30,39]. However, in this work we focus on improving cluster throughput by managing the capacity at the main memory level. Prior art in this area can be divided into three main categories:

Modifying the Memory Hierarchy to Hide/Avoid Disk Activity. Several *high-cost* proposals argue for the need to redesign the memory hierarchy [29,31,9], or add additional resources to the cluster in order to avoid prohibitive disk activity. In particular, a recent proposal; the RAMCloud [25], motivates the need to replace disk storage with permanent RAM storage in order to reduce latency and improve throughput. The RAMCloud requires thousands of dedicated, interconnected commodity servers attached to the cluster to deliver its promise, which as the authors mention in their paper, comes at a high cost per bit, and high energy usage per bit.

In [18], Lim *et. al.,* avoid to over-provisioning individual servers by encapsulating large portion of memory in remote dedicated memory blades which dynamically assign memory to individual servers when needed. Although this scheme provides better utilization of aggregate memory resources, it is targeted for commodity blade servers and may require hardware changes to access the remote memory.

Management of Memory Resources under Single OS Image. In distributed systems with a single OS image (DOSes) [40], the entire address space is made visible to each process running on any node. This assumes a coherent underlining shared memory architecture. Several previous studies have shown that DOSes suffer performance and scalability [3] issues due to their shared memory architecture. Further, as reported in [3], DOSes are relatively expensive to maintain and to deploy.

Management of Memory Resources under Multiple OS Images. Works that belong to this category are closest to our work in terms of the scope of the problem. In distributed systems with multiple OS images, each node in the system can leverage remote memory at another node by either paging to/from the remote memory [17,24,19,20,43,6], or by extending its address space to encapsulate the remote memory. However, these schemes lack the ability to deal with the temporal/spatial node memory requirements fluctuation within the cluster to achieve optimized performance and energy-efficient memory collaboration. Further, prior proposals do not provide Quality-of-Service measures to protect nodes donating part of their memories from having their performance negatively impacted. To address these concerns, we design a run-time mechanism to manage the memory resources across collaborating nodes within the cluster, and we provide QoS measures for individual nodes.

4 Autonomous Collaborative Memory System: Architecture, Protocol and Algorithm

In this section, we describe the proposed Autonomous Collaborative Memory System (ACMS), including ACMS architecture, protocol and algorithm. We adhere to the following design philosophies while designing our system: low operation overhead, high system stability and QoS guarantees for nodes that donate their memories.

4.1 ACMS Architecture

Figure 2 shows a high level ACMS architecture, which consists of the following components.

1. Interconnect. The interconnect medium used to link cluster nodes with each other. We do not specify strict requirements on the type of the interconnect. Although we conduct our prototype and analysis over Ethernet, the ACMS interconnect could be as well PCIe, Lightpeak (Thunderbolt) [13], Infiniband [17], etc.
2. Collaborating Nodes. These represent individual computing nodes comprising the cluster. The nodes may use remote memory (i.e., memory clients), provide memory for other nodes(i.e., memory servers), or neither (i.e., memory neutrals). (Detailed dissuasion in 4.2)
3. Collaborative Memory Service Manager. The manager, with the proposed protocol and algorithm, is responsible for memory discovery, memory allocation and release. The service manager could be a centralized manager responsible for managing all nodes in the cluster, or distributed across all nodes or a collection of nodes. In this paper, we propose a fully distributed memory acquisition protocol that does not require centralized control. Each node makes its decision of when, and with whom it shall collaborate.

It's worth noting that although we focus on remote memory swapping in this paper, the ACMS protocol and algorithm can also be applied to other remote memory leverage approaches such as remote memory mapping.

4.2 Node Classification Algorithm

As mentioned in Section 2, static memory collaboration lacks the desired performance due to typical cluster variations. It is important to dynamically discover, allocate and reclaim remote memory adapting to the nodes condition, to optimize the whole cluster performance and energy efficiency.

To this end, first we classify nodes into three main categories according to their run-time memory usage:

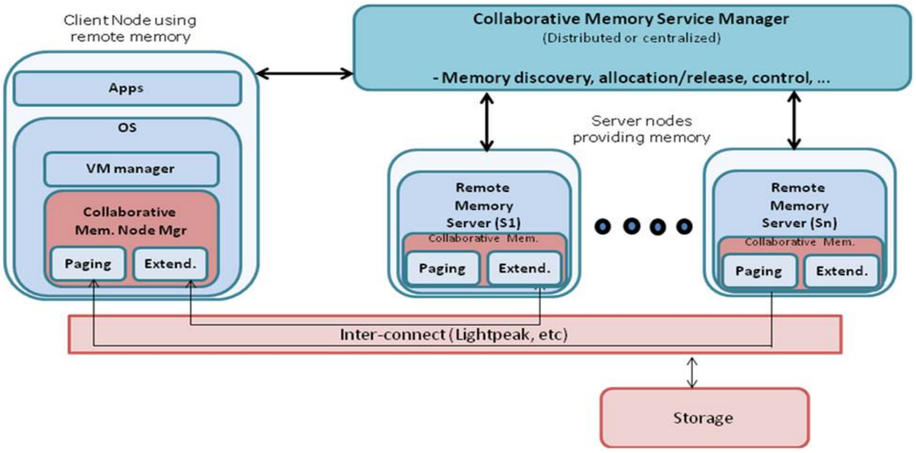

Client Node using remote memory
Apps
OS
VM manager
Collaborative Mem. Node Mgr
Paging
Extend.
Collaborative Memory Service Manager
(Distributed or centralized)
- Memory discovery, allocation/release, control, ...
Server nodes providing memory
Remote Memory Server (S1)
Collaborative Mem.
Paging
Extend.
Remote Memory Server (Sn)
Collaborative Mem.
Paging
Extend.
Inter-connect (Lightpeak, etc)
Storage

Fig. 2. ACMS High-level Architecture

1. A memory client node: a node that is running a high demand application and needs extra memory space.
2. A memory server node: a node that possesses significant amount of free memory and can potentially donate part of its memory space to remote memory clients.
3. A memory neutral node: a self-satisfied node that has mid-level memory usage that neither offers memory nor needs extra memory.

In general, when the memory usage is smaller than MEM_MIN (indicating very low local memory demand), the node is classified as a memory server; if memory usage is larger than MEM_MAX (indicating very high local memory demand), the node becomes a memory client; on the other hand, if memory usage stays between MEM_MIN and MEM_MAX, the node is a neutral node that is self-satisfied. In our classification algorithm, guard bands are applied to both MEM_MIN and MEM_MAX to prevent system oscillation. This attribute is crucial for the stability of the system as it limits nodes oscillation from a memory client to a memory server and vice versa. Specifically, four thresholds, MEM_MIN_LOW, MEM_MIN_HIGH, MEM_MAX_LOW, MEM_MAX_HIGH are used to decide when to change the node classification. When the memory usage is within the "no change" guard bands, no node class change is asserted, as illustrated in Figure 3. The memory thresholds are programmable parameters that give designers the flexibility of fine tuning based on workloads' characteristics.

4.3 Dynamic Memory Acquisition Protocol

During run-time, nodes are classified into their corresponding category, and engage in the ACMS using the memory acquisition protocol described in this section. The proposed protocol allows nodes to exchange information about their

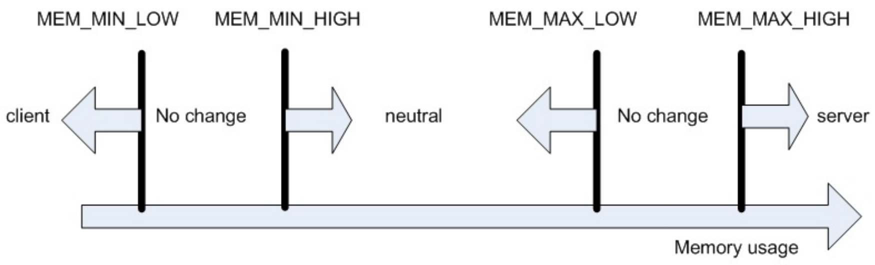


Fig. 3. An illustration of the node classification algorithm showing how memory servers and memory clients are classified. Further, it shows the guard bands used to limit node oscillation.

memory availability, and facilitate dynamic memory collaboration decision in a distributed fashion.

There are five primitives defined for the protocol, as described below.

1. OFFERING_MEM: This message is periodically broadcast by a memory server to all other nodes in the system to indicate its memory availability. The message includes the ID of the memory server, and the amount of available memory. In ACMS, we also monitor the variation of the available memory. If available memory stays relatively stable with little variation, the broadcast frequency is reduced accordingly to reduce the operation overhead without impacting the freshness of the information.
2. REQUESTING_MEM: This message, generated by a memory client is either broadcast to all the other nodes, or sent out to one or more memory servers, responding to a previous OFFERING_MEM message. In this message, the client indicates that it requests free remote memory. In the case that a memory client has multiple potential memory servers to choose from, the client selects a subset of servers based on certain criteria and arbitration mechanism, for example, First Come First Serve (FCFS) for simplicity, Round Robin (RR) for fairness, Nearest Client First (NCF) for more energy efficient collaboration, etc.[1]. One interesting future direction is how to select appropriate memory servers to optimize whole cluster performance and energy efficiency considering node idle/active state.
3. GRANTING_MEM: This message is sent out by a memory server to a given memory client responding to a REQUESTING_MEM message. Note that, this does not bind a memory server with a memory client since the client may get multiple grant messages from multiple servers.
4. ACK_MEM: This message is sent by a memory client to one and only one memory server responding to a GRANTING_MEM message. This message

[1] In our implementation we consider FCFS arbitration scheme. However, optimizations based on other arbitration schemes, topology, real-time network traffic are left as future work.

binds a memory client with a memory server. The client may have to do some arbitration to select one of the servers that granted memory. ACK_MEM message indicates the end of a handshaking transaction to bind a certain memory server with a memory client.

5. RECLAIM_MEM: In order to provide guarantees that a memory server does not get hurt by much when it engages in memory collaboration, we give the memory server the opportunity to stop donating its memory when deemed necessary. To achieve that, when the memory server's memory needs change and gets classified as a memory neutral, it sends a reclaim message to the remote client using its memory to reclaim its granted memory. Once the remote client receives this message, it starts migrating its data back from the remote server.

In order to reduce message broadcasting overhead in the system, we monitor the ratio of memory clients to memory servers during run-time, and the responsibility of broadcasting could be offloaded to the group with the smaller number of nodes. For example, in a network environment heavy with memory servers, it is more appealing to let "few" memory clients broadcast their memory needs, instead of letting "many" memory servers broadcast their memory availability, which leads to higher operation overhead.

Depending on who initiates the broadcast message, the memory acquisition process consists of either a 3-way handshake protocol or a 4-way handshake protocol, as illustrated in Figure 4.

4.4 Discussion

Memory Usage Monitoring. Memory usage can be monitored by either software or hardware approaches. In our prototype (described in next Section), we use OS counters to monitor the memory usage such as MemTotal, MemFree, Buffers, Cached, etc. However, for hardware-based memory collaboration, memory monitoring could be done via hardware techniques such as installing a Memory Monitoring Circuit (MMON) [28]. MMON uses the classical stack distance histogram (SDH) analysis [38] to estimate the memory requirements (e.g., page miss rate, etc.) at run time. It is worth noting that current disk swapping is an OS functionality. However, in Section 6, we discuss the option of having a hardware accelerator to achieve very fast disk swapping. In such cases, hardware-based memory monitoring becomes crucial for optimal performance.

Memory Collaboration Scalability. We discussed the broadcast-based memory acquisition protocol, which we implement and evaluate in a small-scale cluster prototype (5 nodes) in the coming sections. However, when the scale of the system grows to tens, hundreds, or even thousands of nodes [15], the scalability characteristics must be taken into consideration.

In this subsection, we discuss how the proposed protocol scales with larger cluster systems. We propose a *Zone-Based Broadcast Protocol* (ZBBP), which limits each node to only broadcast the messages to its n-hop neighbors. The broadcast and memory sharing scope is limited due to two main reasons:

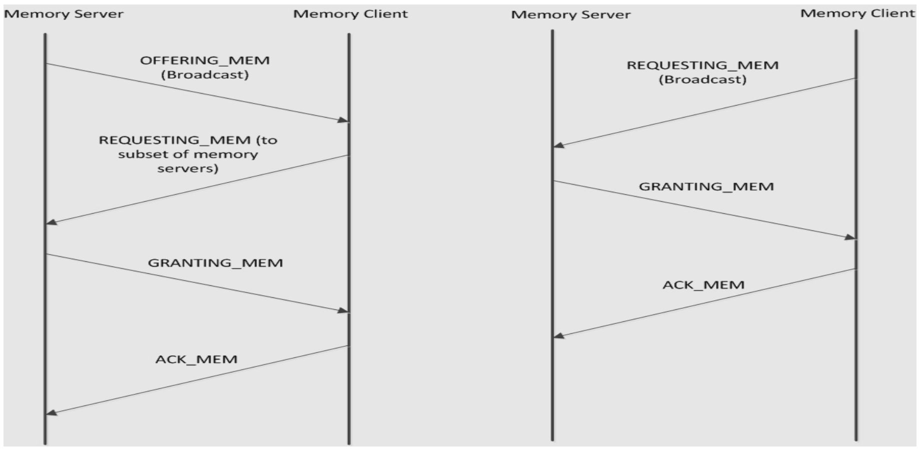


Fig. 4. Protocol illustration: (Left) 4-way handshake if server initiates broadcast, (Right) 3-way handshake if client initiates broadcast

1. First, it reduces the broadcast overhead – processing time, interconnect traffic which leads to reduction of interconnect power. A node can only broadcast to its n-hop neighbors instead of the whole network. Hence, the overhead of processing the broadcast messages and the burden on the network are greatly reduced. Due to this, the overall overhead of collaborative memory is reduced as well.
2. Second, it improves distance locality. Forcing a node to only share memory within its n-hop neighbors instead of sharing memory with nodes located far away is important for both performance and energy considerations. Accessing close-by nodes incurs less latency, both due to smaller number of hops and also less chance to encounter congestion.

Thy ZBBP works operates as follows. When a node broadcasts its memory availability to its n-hop neighboring nodes, it incorporates the maximum hop count; *max_hop* as a parameter within the message. The maximum hop count is initially set to the radius of the zone. For example, a maximum hop count is set to 3 if a node is allowed to only share memory with other nodes at most 3 hops away. The nodes who receive the message will continue to process the message according to the discussion in Section 4.3, however, with a slight modification. The node will extract the hop count from the message, if the hop count is greater than zero, it decreases the hop count by one and forward the message to all its neighbors except the one from whom it received the message. If the hop count is zero, the node processes the message without forwarding it any further. If a node receives the same message for the second time, it will discard the message without processing/forwarding it to reduce broadcast overhead. Additional methods, such as, Smart Gossip [16] can be applied to further reduce broadcast overhead. However, discussing these works is outside the scope of this paper.

Memory Collaboration Reliability. When performing remote memory collaboration in clustered systems, reliability is an important issue to consider. Any failure in remote paging handling would degrade system stability and even result in crashing the system. In our design, TCP/IP or RDMA is used as network transport protocols. Since they are both reliable protocols, the message correctness is guaranteed. other aspects of reliability, e.g., protection against crashed remote memory servers, are discussed in [19]. Reliability techniques such as mirroring and parity can be applied as well to maintain normal operation in the case of memory server crash. We do not elaborately discuss these issues in this paper.

Next we will discuss the prototyping for our Autonomous Collaborative Memory System. Although in our exemplary implementation we consider FCFS arbitration scheme to select a remote client/server, one might utilize other schemes to deliver optimal energy efficiency or better fairness. Further optimizations based on topology, real-time network traffic and workloads for energy-efficiency are left as future work.

5 System Implementation and Prototyping

In this section, we describe the system implementation of the proposed ACMS to conduct feasibility and benefit, and to quantify and evaluate the overhead of remote memory paging. For prototyping purposes, we make the following three design choices. (1) We leverage remote memory by applying remote memory swapping (as opposed to remote memory mapping). One main reason, as we also mentioned in Section 3, is that swapping requires less system modification and provides a feasible and rapid implementation approach to study and analyze ACMS performance and bottlenecks 7.

(2) We choose Ethernet as the interconnect among the computing nodes and use TCP/IP suite as the communication protocol for inter-node communication. However, our protocols can be also implemented over other types of interconnects and communication protocols, for example, Remote DMA access (RDMA) over infiniband [17], lightpeak [13], and PCIe.

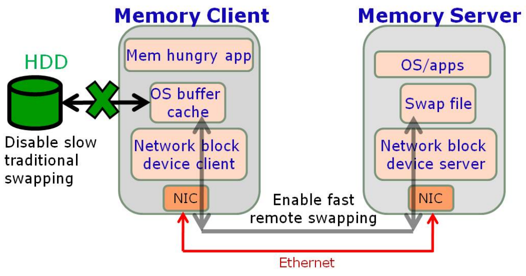

Memory Client
Memory Server
HDD
Mem hungry app
OS buffer cache
Network block device client
NIC
Disable slow traditional swapping
OS/apps
Swap file
Network block device server
NIC
Enable fast remote swapping
Ethernet

Fig. 5. An example of a collaborative memory system consisting of two nodes, a memory client, and a memory server

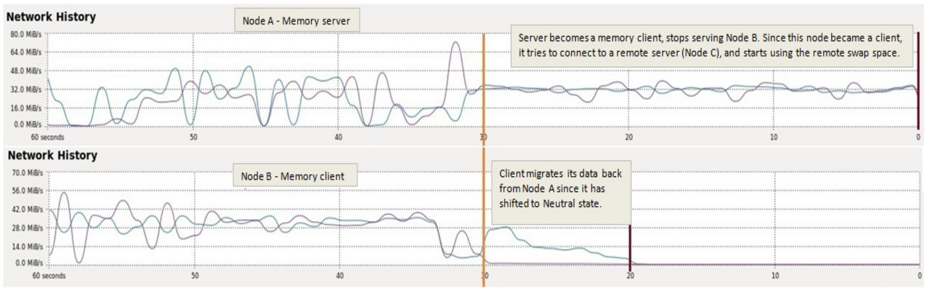


Fig. 6. ACMS dynamic memory collaboration activity represented by network traffic monitoring. An injected run-time change at about the 30th second, shifts Node A to become a memory client, and Node B to become a memory Neutral.

(3) We implement the dynamic ACMS memory detecting, allocation and de-allocation protocol as a process running in user space. As a result, no kernel or application modification is required.

A memory server node donates a portion of its memory space to a remote memory client. Once the memory client runs short on its local memory, it disables local, slow hard disk swapping, or assigns it a low priority. At the same time, it enables remote swapping as shown in Figure 5.

In order to facilitate swapping over Ethernet, we have leveraged several extant features in current operating system kernels. Among them is an external kernel module called Network Block Device (NBD) [23].[2] Once setup over network, NBD allows the local file system at the memory client to access a remote file system at the memory server transparently, hence, adding the ability to swap remotely. Further, the local swap device (i.e., HDD) can be assigned lower priority via *swapon/swapoff* system calls.

The node classification algorithm, as well as the dynamic memory acquisition protocol (discussed in Section 4.2, Section 4.3), are implemented as user-space threads at each node. This allows each node to dynamically identify run time memory usage and communicate information with other nodes to accomplish ACMS objectives.

6 System Performance Evaluation

In this section, we evaluate the performance of the proposed ACMS comparing to a traditional system with Hard Disk Drive (HDD) swapping, a system with

[2] An NBD device consists of two components complementing each other, an NBD-client and an NBD-server. In order for our remote paging to operate through an NBD device, each memory server has to create a swap partition in its memory (i.e a regular swap file, or a RAM disk). Then it attaches the swap partition to its NBD-server. At the same time, each memory client establishes a connection between the NBD-client daemon at the memory client side, and the NBD-server at a memory server side.

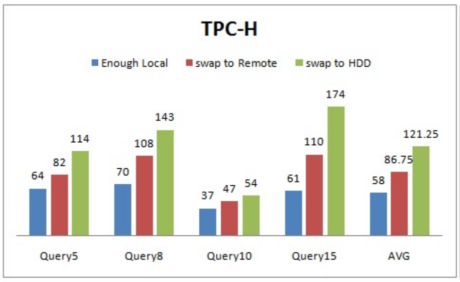
Solid State Drive (SSD) swapping, to a system with static memory collaboration, and to a system with enough local memory. The summary is that ACMS can significantly improve the memory client performance (up to 3x) without perceivable performance impact on memory servers compared to HDD swapping, and performs on par with SSD swapping.

Our experimental setup consists of multiple (up to 5) 2.6GHz Intel ® Core™ i7-based machines [14], each one having 4GB of RAM, and a 250GB 7200RPM HDD. Machines are running a Fedora 14 [10] OS with a Linux kernel version 2.6.35, and are connected via 1Gbps Ethernet NICs. Further, we are using a network block device version 2.9.23. In order to control the amount of available memory available at the local node to study system behavior under different memory provision and usage conditions, we have developed a memory *balloon* that inflates and locks a user-specified amount of local memory. To test and analyze the system behavior, we use both micro benchmarks we developed for controlled environment analysis, as well as real-world applications such as SPEC CPU2006 [37], TPC-H [41] with PostgreSQL 9.0.4 DataBase, Apache Hadoop [2], and SPECjbb [36].

Dynamic Behaviour. Figure 6 shows the autonomous operation of our ACMS dynamics. The figure shows the network traffic at two nodes, a memory server node A (top), and a memory client node B (bottom). The left half of both figures shows the traffic while nodes A and B collaborate with each other (i.e., A is servicing B). At around the 30th second, an application with large memory demand starts on node A, meanwhile memory demand on node B decreases gradually. This run-time change causes A to become a memory client and B a memory neutral. As a result, and as described in Section 4, A sends out a reclaim message to node B to reclaim its memory back. Once B receives the message, and starts migrating its data back which lasts for about 10 seconds.[3] Meanwhile, A starts collaborating with a third node C (not shown in the figure) with A acting as a memory client. The traffic at the right portion of the figure shows the swapping activity being sent out to node C. At the same time, node B becomes a neutral that does not collaborate with remote nodes. This visual illustration shows the dynamics and elasticity of the ACMS protocol given the changing memory requirements for running workloads.

Performance Comparison against Local Memory and HDD. Figure 7 shows the application performance (as measured by completion time in seconds) for various TPC-H queries, and for sorting various data structure sizes using Hadoop workloads. These experiments are conducted using two machines only with one acting as a memory server, and the other acting as a memory client, with the configurations mentioned in Section 6. The legends in the figures represent the completion time while running on a system with *enough* (4GBs free) local memory, a system with *limited* (less than 200MBs free) local memory and

[3] The time it takes for a client to migrate its data back depends on the network speed (e.g., 1Gbps, 10Gbps, etc.) and the amount of data that resides on the remote swap space.



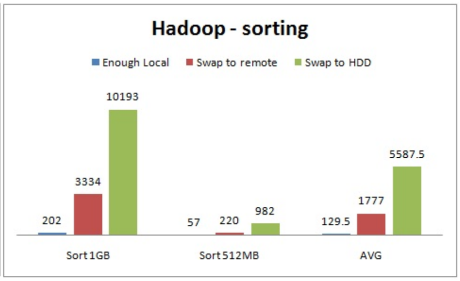


Fig. 7. Performance of TPC-H and Hadoop while running with enough local memory, limited local memory/swapping to remote memory, and limited memory/swapping to HDD

swapping to remote memory, and a system with *limited* (less than 200MBs) local memory and swapping to HDD, respectively. As shown in the figures, swapping to remote memory can improve the performance by an average speedup of 1.4x in TPC-H and 3.1x in Hadoop.

The reason why TPC-H provides less performance improvement compared to Hadoop is that, TPC-H is optimized to operate within the available local memory. If the system has limited local memory, TPC-H is optimized to reduce its memory footprint in order to limit disk swapping. On the other hand, Hadoop does not pay similar attention to the available local memory, thus resorting to swapping more frequently. This shows that having an efficient swap device could potentially reduce the programming effort needed to optimize workloads such as TPC-H.

The figures also show that running with enough local memory renders much better performance compared to remote swapping, which is expected. Accessing data in the local main memory is faster than accessing data in a remote memory space, due to both the network latency and the swapping overhead (discussed in more detail in section 7).

Thus far, we have shown that remote swapping improves performance for memory clients. However, this performance improvement should not come at the expense of performance degradation for memory servers. Figure 8 shows the completion time for several memory intensive SPECCPU2006 applications running on memory servers. The results show that the applications' performance degraded very little, confirming the resilience of memory servers to memory collaboration. This robust behavior is a result of the ACMS adaptive design which can detect the lack of memory at the memory server side and signal remote clients to relinquish their allocated portions and migrate their data back. Other workloads such as TPC-H, and Hadoop show similar trends to the SPECCPU2006 benchmarks, hence, we omit these figures.

Performance Comparison against SSD. Solid State Drives (SSDs) present a promising technology to replace HDDs since they offer an order of magnitude lower access time latency and more reliable structure as they don't

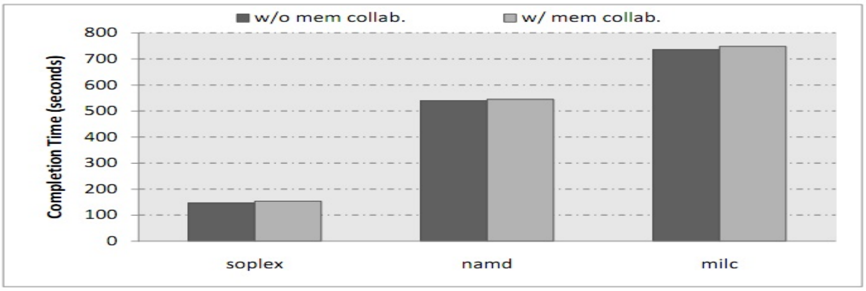


Fig. 8. Impact on memory servers while running several SPECCPU2006 applications

include spinning disks and movable read/write heads, however at a much higher price. Figure 9 shows the performance of SPECjbb as measured by the number of instructions retired per unit time. The figure compares the performance of SPECjbb while running with enough local memory, swapping to remote memory, and swapping to SSD. The x-axis in the figure shows the run-time in seconds. As can be seen, on average, swapping to remote memory performs on par with swapping to SSD device. The figure also shows that both schemes fall short behind running with enough local memory.

In summary, memory collaboration achieves an average speedup of 3x compared to a non-collaborative memory system, performs on par with SSDs and falls short behind running with enough local memory.

7 Remote Swapping Overhead Analysis

As shown earlier, remote memory swapping achieves significant speedups compared to traditional disk swapping, and performs on par with SSD swapping. However, the performance of remote memory swapping also falls way short compared to running the application entirely on local memory, even with the consideration of interconnect propagation delay, which is a physical limitation. In this

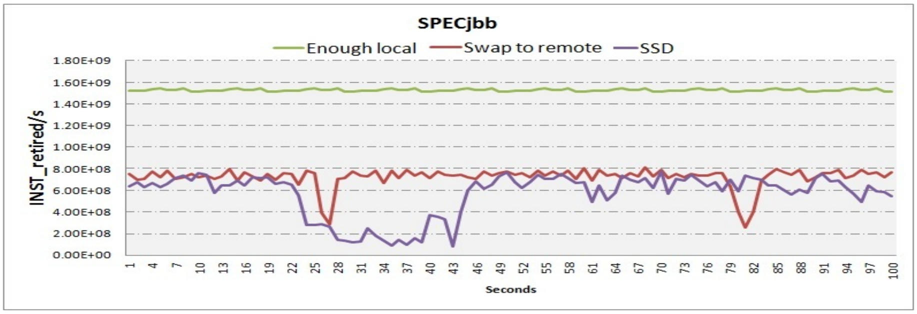


Fig. 9. Performance of SPECjbb as measured by the number of instructions retired for swap to remote, swapping to SSD, and using local Memory

section, we investigate the timeline of remote swapping and potential overhead sources. The high level summary of the analysis is that the network stack and kernel swapping handling process are two major sources of low performance. Based on the understanding, we provide insights on how to reduce the overhead and improve performance.

CPU Load Consideration. In our prototype, all processing, both on the client and server side, is done by the host processor. There are no special hardware accelerators (e.g., remote DMA engine) that handle portions of the processing. However, our system profiling has shown that CPU is idling more than 70%-80% of the time waiting for I/O requests to complete. *This shows that CPUs are not overloaded.*

Network Bandwidth Consideration. We conducted our experiments over 1Gbps Ethernet links between clients and servers. Our network monitoring tools confirmed that only about 50% of the network bandwidth is being utilized. In today's data center and cluster system, usually 10Gbps Ethernet links are not uncommon. Other Interconnects such as Lightpeak [13] has significantly higher physical bandwidth. *Hence, network bandwidth is not a main bottleneck*, at least before other bottlenecks are removed.

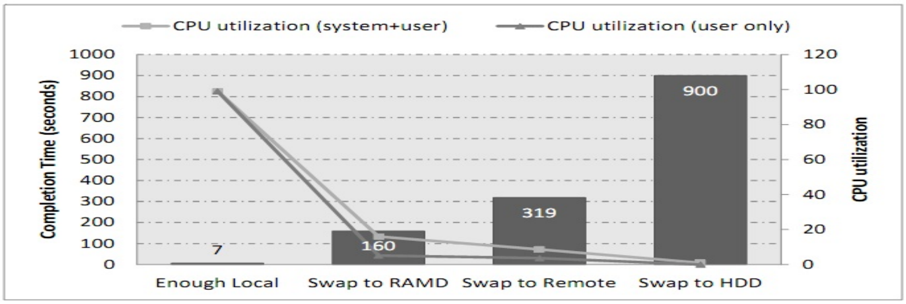


Fig. 10. Completion time and CPU utilization for various swapping schemes

Network Stack and Swapping Overhead. In our prototype, all communications between nodes go through the TCP/IP stack and an NBD device, making them potential major bottlenecks. In order to show the impact of network stack and the NBD device, we conducted the following experiment. We created a RAMDisk as a local swap device on the local memory itself. When the system runs short on memory, it starts swapping to/from the local RAMDisk. This operation does not involve any TCP overhead or NBD device overhead since the swap device is located locally. Figure 10 shows the completion time for a micro benchmark application while running with enough local memory, limited local memory/swapping to local RAMDisk, limited local memory/swapping to remote machine over network, limited local memory/swapping to local disk. The figure shows two interesting observations.

First, avoiding the network delay including TCP/IP stack, NBD device operation and propagation delay, can save almost 50% of the overhead (319sec to 160sec). Considering the very small propagation delay (on the order of a few micro-seconds), the network stack proves to be a major bottleneck.

Second, even though the RAMDisk is located locally (no network involved), swapping to RAMDisk still performs much worse than running with enough local memory. The reason for that will become clear if we look at the top curved line in the same figure which shows the CPU utilization of the running application.[4] The CPU utilization is 100% when the application runs entirely on local memory, 20% while swapping to RAMDisk, 8% while swapping to remote memory, and less than 1% while swapping to local disk. The bottom curved line represents the CPU utilization while executing user-space code only (i.e., excluding system CPU utilization), which shows that even those modest CPU utilization numbers do not correspond to useful work all the time. Thus, kernel swapping proves to be another major bottleneck.

Next, we are going to discuss the network and kernel swapping overhead in details.

7.1 Network Overhead Analysis

In this subsection, we investigate the overhead induced by accessing remote memory through the network over TCP/IP stack. We provide an overview on the life cycle of bringing a page from remote memory into local memory in order to understand the cost of network related operations.

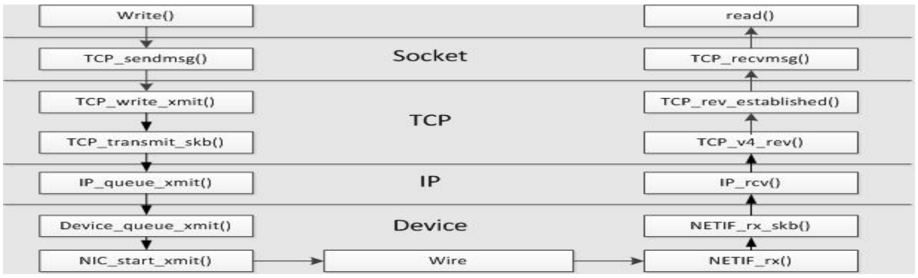


Fig. 11. Life cycle of a packet over the network

Figure 11 shows the end-to-end life cycle of a packet from the sender to the receiver. For the sender side, the processing begins when the NBD is called to send a packet over the network. The packet is copied into the TCP socket for TCP layer processing, e.g., checking the socket's write queue and building TCP header. *TCP_transmit_skb()* then sends the packet to IP layer for IP processing.

[4] CPU utilization is measured as (CPU time executing user space code (userTime) + CPU time executing system code (systemTime))/Wall clock time.

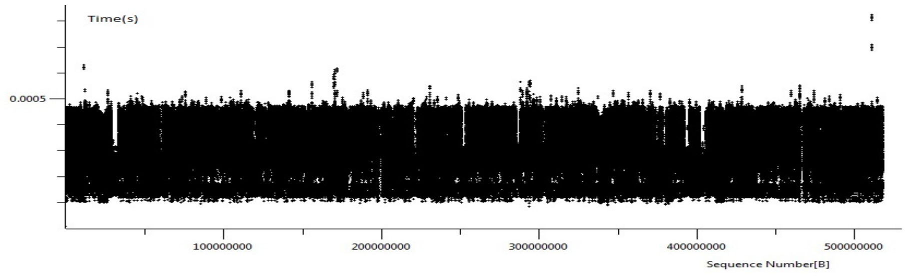


Fig. 12. Measured Round Trip Time (RTT) using 1Gbps Ethernet over local network

Finally the IP datagram is put into device queue, and the host signals the Network Interface Card (NIC) to transmit. Upon receiving the signal (called door bell), NIC DMAs the packet from the host, performs necessary MAC processing, sends the packet over the wire using a pre-configured interface, and updates the appropriate descriptors.

Similarly, on the receiver side, NIC receives a frame from the wire and extracts the TCP/IP packets by removing the frame delineation bits. NIC also performs CRC check sum to confirm the correctness of the packet. The NIC then grabs the next available descriptor, which indicates the memory location for the NIC to copy the packet to. the NIC DMAs the packet to the pre-allocated buffer and interrupts the host for packet processing. The packet travels through IP layer processing, TCP layer processing and finally reaches the destination through socket buffer.

Due to various processing delays in the stack and the NIC, the *average* Round Trip Time (RTT) for a packet can reach 250us for 1Gbps Ethernet card on client machines or low end servers, as we measured on our system. (Figure 12 shows the scatter-plot of packet RTTs for a TCP connection.)

Comparing the latency of retrieving pages from remote memory over unoptimized network stack to the 60ns local memory access latency justifies, partially, the long CPU I/O waiting times and low CPU utilization.

7.2 Kernel Swapping Overhead Analysis

As mentioned earlier, even if the network overhead is eliminated (RAMDisk case), the swapping approach does not perform well compared to running on the local memory. There are several reasons why the CPU utilization is low when applications resort to swapping. We summarize these issues into the following.

1. When a page fault occurs, an exception is raised followed by a CPU pipeline flush in order to service the page fault routine. Pipelining is a powerful technique to hide the long memory latency. Flushing the pipeline frequently reduces the effectiveness of latency hiding, hence rendering a low CPU utilization. Further, executing the page fault routine pollutes the data and instruction caches, TLBs, and key processor structures. Prior work [35] shows

that if for example, SPEC-JBB throws an exception once every 1k instruction, its performance could be degraded by up to 48% due to the aforementioned reasons.
2. When a page fault occurs, the OS scheduler assumes that the page fault is going to take long time to finish, hence, it context-switches the process out and adds it to the I/O waiting queue. This adds a fixed overhead to every page fault regardless of how fast it gets serviced.
3. If the memory pressure is very high, the OS blocks the running process until the kernel swap daemon (AKA *kswapd*) frees some memory. This scenario is known as congestion wait. Our kernel probing and profiling of the kswapd indicates that the function *get_swap_page* - which is used to find a potential contiguous space in the swap out device to allocate swapped-out pages, consumes more than 45% of the system CPU clock cycles, and more than 74% of the retired instructions.
4. Under high memory pressure, the kernel performs heavy page scanning to figure out which page is next to be replaced (or swapped).
5. When the system has to free pages, some clean pages get dropped from the page cache. These clean pages may correspond to the *program code* that is already running. In which case, the OS has to bring them back as the program continues execution.

Therefore, once the system resorts to swapping, regardless of how fast or optimized the swap device is (remote memory, SSD, etc.), the system performance degrades significantly due to the inherent limitation in kernel swapping method which is designed for very slow devices such as the HDD.

7.3 Remote Swapping Optimizations

We discuss several directions in both software and hardware space to reduce the remote swapping latency and improve the CPU utilization and overall system performance.

Network Optimizations

1. By applying several network optimizations such as turning off the Nagle Algorithm, TCP delayed ACK [26], and enabling Direct Cache Access or Direct IO [12], the average RTT for a packet could be reduced from 250μs to around 22μs or even lower. Table 1 shows the measured RTT on a high end server system with different commercial Ethernet NICs. With smaller RTT, the application runtime can be reduced as well.
2. Reducing TCP/IP processing overhead and the data copy latencies by using a special interconnect with low latency, and protocols that bypass the traditional TCP/IP stack. One leading technology is InfiniBand [17,11] (IB) which provides very low latency and high throughput (40Gbps). Native IB verbs form the lowest software layer and allow Remote Direct Memory (RDMA) between local memory and remote memory while bypassing the operating system. Using full-duplex quad data rate (QDR) IB of 40Gbps, we measured

Table 1. Measured RTT on a server platform

Mellanox	Intel	Broadcom	Solarflare
12μs	16.8μs	21.6μs	7.4μs

the average packet RTT to go below 5μs. Even with Ethernet NICs, bypassing default TCP/IP stack can also reduce the latency. The lowest measured latency on Mellanox IB NIC with RDMA is about 1.3μs, while lowest latency of the Ethernet counterpart (with RDMA) is measured at about 1.6μs. This potentially will improve the performance significantly.

Kernel Swapping Optimizations

1. It is crucial to limit the severe performance impact of exceptions raised by page faults in order to reduce the drop in CPU utilization. To achieve that, the OS can leverage multi-core processors by servicing the page-fault on one of the idle core [35]. This potentially prevents CPU pipeline flushing and TLB, data, and instruction cache pollution, hence, rendering higher CPU utilization and better performance.
2. The virtual memory management attributes can be manipulated based on the swap device characteristics. Such attributes includes: (1) the number of pages to fetch at a time when the swap device is accessed (i.e., prefetching). (2) the time quantum at which the OS starts freeing dirty pages from memory. (3) the threshold at which the OS consider its free memory to be low.
3. The Linux kernel attempts at reducing the latency of the HDD's head spinning by having the I/O elevator rearrange and aggregate I/O requests that belong to the same disk sector. However, with fast swap devices such as remote memories and SSDs, there is no head spinning involved, hence, there is no need to stall I/O requests trying to aggregate them. We set the I/O elevator to *noop* which issues I/O requests to the swap device as they are ready.
4. Having a hardware accelerator to intercept page fault interrupts and service them in hardware, thus effectively avoiding kernel context switches. The accelerator manages the page table for swapping and establishes a link with a remote memory server to facilitate efficient swapping. Using this approach, the page swapping does not have to be limited to page boundary. Instead, the size of swapped data can be adaptively adjusted based on application requirements. Although this optimization could potentially render the best performance, it also comes with the requirement for hardware modification, as well as limited OS modifications (for page table synchronization).

We have evaluated the prefetching optimization and applied it to our ACMS system. By default, the Linux kernel fetches 8 pages from the swap device when servicing a page fault, in an attempt to reduce swap device accesses. We modified this default parameter over a logarithmic range from 1 to 1024 pages. For our micro benchmark, we found that performance improves up to 512 pages, beyond which prefetching starts hurting the application. Figure 13 shows the performance of the same micro benchmark before and after applying the 512-page prefetching optimization. This experiment shows that appropriate prefetching improves this micro benchmark performance by about 25% over unoptimized remote swapping. Note that, since we know the access pattern of the micro benchmark, which is sequential in our case, then we could expect to see benefit by applying the page prefetching technique. However, with more complicated or hard-to-predict access patterns, prefetching may not be a fruitful optimization. This could be done dynamically by monitoring the application's behavior (e.g., using SDH), and apply the prefetching optimization if a sequential access pattern is observed.

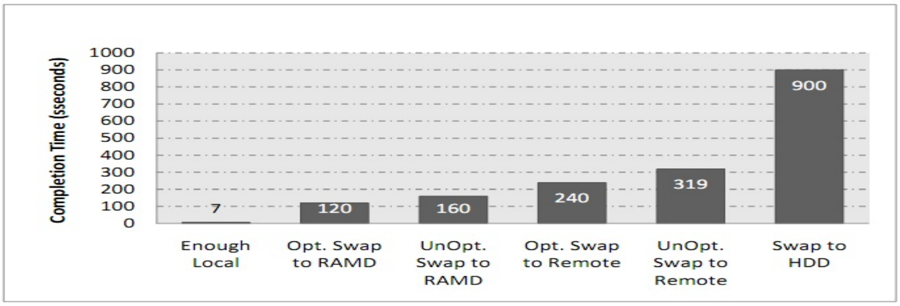


Fig. 13. Completion time for various swapping schemes given the page prefetching optimization

8 Conclusions

Memory collaboration reduces capacity fragmentation in clustered architectures; it allows nodes that need additional memory space to place their data in remote memories instead of slow storage. Current memory collaboration mechanisms lack the ability to provide autonomous memory collaboration and to adapt dynamically with oscillating memory needs by various applications. Further, in order to optimize the performance of memory collaboration, detailed understanding of the major performance bottlenecks in the end-to-end memory collaboration is necessary.

To address these issues, in this paper, we have developed an Autonomous Collaborative Memory System (ACMS) that permits dynamic, run time memory collaboration and provides QoS guarantees for memory servers, i.e., nodes that donate their memory for remote use. We have implemented a prototype realizing the ACMS and our results show up to 3x performance speedup compared to non-collaborative memory system. Further, we conduct a detailed analysis to identify several memory collaboration bottlenecks, and provide insights as to how to overcome such bottlenecks to further improve memory collaboration performance. Our investigation shows that network stack and kernel swapper are the major performance bottlenecks. In the future, we will investigate a detailed implementation and evaluation of the hardware-assisted paging in an attempt to deliver better remote memory access latency that gets close to the latency of the local memory.

9 Future Work

We are investigating the scalability of our ACMS for large scale clusters, e.g., seamicro's new system with more than 1000 nodes. In such systems, there are many interesting questions to be answered, such as: how nodes should communicate efficiently? how to choose memory severs/memory clients for optimized cluster energy consumption? etc.

Acknowledgments. We would like to thank all anonymous reviewers for their helpful feedback and suggestions. In particular, we would like to extend special thanks to: Feng Chen and Nezih Yigitbasi for facilitating TPC-H and Hadoop workload setups; Phil Cayton, Frank Berry, and Michael Mesnier for helping understanding storage and swapping activities; Arjan Van De Ven for helping understanding the kernel swapping mechanism; Eugene Gorbatov for providing several insights on data center provisioning; James Tsai, Mesut Ergin, Rajesh Sankaran, Sanjay Kumar, and Richard Uhlig for the insightful comments and discussions on memory collaboration; Alexander Min and Sameh Gobriel for helping improving the presentation of the paper.

References

1. Agarwal, A.: Facebook: Science and the social graph (2009), `http://www.infoq.com/presentations/Facebook-Software-Stack`; Presented in QCon San Francisco
2. Apache: Hadoop (2011), `http://hadoop.apache.org/`
3. Baumann, A., Barham, P., Dagand, P.E., Harris, T., Isaacs, R., Peter, S., Roscoe, T., Schuepbach, A., Singhania, A.: The multikernel: a new OS architecture for scalable multicore systems. In: SOSP 2009: Proceedings of the 22nd ACM Symposium on Operating Systems Principles. ACM Press, New York (2009)
4. Beckmann, B.M., Marty, M.R., Wood, D.A.: ASR: Adaptive Selective Replication for CMP Caches. In: MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society (2006), `http://dx.doi.org/10.1109/MICRO.2006.10`
5. Chang, J., Sohi, G.S.: Cooperative Caching for Chip Multiprocessors. In: 33rd International Symposium on Computer Architecture, ISCA, 2006 (2006), `http://dx.doi.org/10.1109/ISCA.2006.17`, doi:10.1109/ISCA.2006.17

6. Chen, H., Luo, Y., Wang, X., Zhang, B., Sun, Y., Wang, Z.: A transparent remote paging model for virtual machines (2008)
7. Chishti, Z., Powell, M.D., Vijaykumar., T.N.: Optimizing Replication, Communication and Capacity Allocation in CMPs. In: The 32th ISCA (June 2005)
8. Corp., I.: Chip shot: Intel outlines low-power micro server strategy (2011)
9. Dhiman, G., Ayoub, R., Rosing, T.: PDRAM: a hybrid PRAM and DRAM main memory system. In: Proceedings of the 46th Annual Design Automation Conference, DAC 2009, pp. 469–664. ACM, New York (2009), doi: `http://doi.acm.org/10.1145/1629911.1630086`
10. Fedora Project: Intel. Core. i7-800 Processor Series (2010), `http://fedoraproject.org/`
11. Grant, R., Balaji, P., Afsahi, A.: A study of hardware assisted ip over infiniband and its impact on enterprise data center performance. In: 2010 IEEE International Symposium on Performance Analysis of Systems Software (ISPASS), pp. 144–153 (2010), doi:10.1109/ISPASS.2010.5452035
12. Huggahalli, R., Iyer, R., Tetrick, S.: Direct cache access for high bandwidth network i/o. In: Proceedings of the 32nd Annual International Symposium on Computer Architecture, ISCA 2005, pp. 50–59. IEEE Computer Society, Washington, DC (2005), `http://dx.doi.org/10.1109/ISCA.2005.23`
13. Intel Corp.: Thunderbolt Technology (2011), `http://www.intel.com/technology/io/thunderbolt/index.htm`
14. Intel Microarchitecture: Intel. Core. i7-800 Processor Series (2010), `http://download.intel.com/products/processor/corei7/319724.pdf`
15. Howard, J., Dighe, S.: A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In: Proceedings of the International Solid-State Circuits Conference (ISCC), ISSCC, 2010 (2010)
16. Kyasanur, P., Choudhury, R.R., Gupta, I.: Smart gossip: An adaptive gossip-based broadcasting service for sensor networks. In: 2006 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), pp. 91–100 (2006), doi:10.1109/MOBHOC.2006.278671
17. Liang, S., Noronha, R., Panda, D.: Swapping to remote memory over InfiniBand: An approach using a high performance network block device. In: IEEE International Cluster Computing, pp. 1–10 (2005), doi: 10.1109/CLUSTR.2005.347050
18. Lim, K., Chang, J., Mudge, T., Ranganathan, P., Reinhardt, S.K., Wenisch, T.F.: Disaggregated memory for expansion and sharing in blade servers. In: Proceedings of the 36th Annual International Symposium on Computer Architecture, ISCA 2009, pp. 267–278. ACM, New York (2009), doi: `http://doi.acm.org/10.1145/1555754.1555789`
19. Markatos, E., Markatos, E.P., Dramitinos, G., Dramitinos, G.: Implementation of a reliable remote memory pager. In: USENIX Annual Technical Conference, pp. 177–190 (1996)
20. Markatos, E.P., Dramitinos, G.: Adding flexibility to a remote memory pager (1996)
21. Massie, M.L., Chun, B.N., Culler, D.E.: The ganglia distributed monitoring system: Design, implementation and experience (2004)
22. Midorikawa, H., Kurokawa, M., Himeno, R., Sato, M.: DLM: A distributed large memory system using remote memory swapping over cluster nodes. In: 2008 IEEE International Conference on Cluster Computing, pp. 268–273 (2008), doi:10.1109/CLUSTR.2008.4663780

23. Network Block Device TCP version: NBD (2011), `http://nbd.sourceforge.net/`
24. Newhall, T., Finney, S., Ganchev, K., Spiegel, M.: Nswap: A network swapping module for linux clusters (2003)
25. Ousterhout, J.K., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Rosenblum, M., Rumble, S.M., Stratmann, E., Stutsman, R.: The case for ramclouds: Scalable high-performance storage entirely in DRAM. In: SIGOPS OSR. Stanford InfoLab (2009), `http://ilpubs.stanford.edu:8090/942/`
26. Peterson, L., Davie, B.: Computer networks, 5th edn. (2011)
27. Qureshi, M.: Adaptive Spill-Receive for Robust High-Performance Caching in CMPs. In: IEEE 15th International Symposium on High Performance Computer Architecture, HPCA (2009), doi:10.1109/HPCA.2009.4798236
28. Qureshi, M.K., Franceschini, M.M., Lastras-Montaño, L.A., Karidis, J.P.: Morphable memory system: a robust architecture for exploiting multi-level phase change memories. SIGARCH Comput. Archit. News 38, 153–162 (2010), doi:`http://doi.acm.org/10.1145/1816038.1815981`
29. Qureshi, M.K., Srinivasan, V., Rivers, J.A.: Scalable high performance main memory system using phase-change memory technology. In: Proceedings of the 36th Annual International Symposium on Computer Architecture, ISCA 2009, pp. 24–33. ACM, New York (2009), doi: `http://doi.acm.org/10.1145/1555754.1555760`
30. Rafique, N., Lim, W.T., Thottethodi, M.: Architectural support for operating system-driven CMP cache management. In: PACT 2006: Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques, ACM (2006), doi: `http://doi.acm.org/10.1145/1152154.1152160`
31. Ramos, L.E., Gorbatov, E., Bianchini, R.: Page placement in hybrid memory systems. In: Proceedings of the International Conference on Supercomputing, ICS 2011, pp. 85–95. ACM, New York (2011), doi: `http://doi.acm.org/10.1145/1995896.1995911`
32. Rao, A.: Seamicro technology overview (2010)
33. Samih, A., Krishna, A., Solihin, Y.: Understanding the limits of capacity sharing in CMP Private Caches, in CMP-MSI (2009)
34. Samih, A., Krishna, A., Solihin, Y.: Evaluating Placement Policies for Managing Capacity Sharing in CMP Architectures with Private Caches. ACM Transactions on Architecture and Code Optimization (TACO) 8(3) (2011)
35. Soares, L., Stumm, M.: Flexsc: flexible system call scheduling with exception-less system calls. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI 2010, pp. 1–8. USENIX Association, Berkeley (2010), `http://dl.acm.org/citation.cfm?id=1924943.1924946`
36. SPEC: SPECjbb2005, `http://www.spec.org/jbb2005/`
37. Standard Performance Evaluation Corporation (2006), `http://www.specbench.org`
38. Suh, G., Devadas, S., Rudolph, L.: A new memory monitoring scheme for memory-aware scheduling and partitioning. In: Proceedings of the Eighth International Symposium on High-Performance Computer Architecture, pp. 117–128 (2002), doi:10.1109/HPCA.2002.995703
39. Tam, D.K., Azimi, R., Soares, L.B., Stumm, M.: RapidMRC: Approximating L2 Miss Rate Curves on Commodity Systems for Online Optimizations. SIGPLAN Not. 44(3) (2009), doi: `http://doi.acm.org/10.1145/1508284.1508259`
40. Tanenbaum, A.S., Van Renesse, R.: Distributed operating systems. ACM Comput. Surv. 17, 419–470 (1985), doi: `http://doi.acm.org/10.1145/6041.6074`
41. Transaction Processing Performance Council: TPC-H 2.14.2 (2011), `http://www.tpc.org/tpch/`

42. vmware : experience game-changing virtual machine mobility, `http://www.vmware.com/products/vmotion/overview.html` (2011)
43. Wang, N., Liu, X., He, J., Han, J., Zhang, L., Xu, Z.: Collaborative memory pool in cluster system. In: International Conference on Parallel Processing, ICPP 2007, p. 17 (2007), doi:10.1109/ICPP.2007.25
44. Zhang, M., Asanovic, K.: Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In: ISCA 2005: Proceedings of the 32nd Annual International Symposium on Computer Architecture, IEEE Computer Society (2005), doi: `http://dx.doi.org/10.1109/ISCA.2005.53`

Multilayer Perceptrons Which Are Tolerant to Multiple Faults and Learnings to Realize Them

Tadayoshi Horita[1], Itsuo Takanami[2], and Kazuhiro Nishimura[1]

[1] Polytecnic University, 2-32-1, Ogawanishimachi, Kodaira-shi, Tokyo 187-0035, Japan
horita@uitec.ac.jp
http://www.uitec.jeed.or.jp/english/index.html
[2] Ichinoseki National College of Technology in Former Times, Iwate-ken, Japan

Abstract. We discuss a fault-tolerance of multilayer perceptrons in which input and output learning examples are patterns consisting of 0s and 1s. A type of faults to be dealt with is a multiple neuron and/or weight fault where neurons are in the hidden layer and weights are between the hidden and output layers. We theoretically analyze the condition when a multilayer perceptron is tolerant to multiple neuron and weight faults. According to the analysis, we propose two value injection methods denoted as VIM-WN and VIM-N to make multilayer perceptrons tolerant to all multiple neuron and/or weight faults whose values are in a multi-dimensional interval. In VIM-WN, the extreme values specified by the fault ranges are set to the outputs of the selected neurons and the selected weights of the links at the same time in a learning phase. In VIM-N, the extreme values specified by the fault ranges are set only to the outputs of the selected neurons likewise. First, we present an algorithm based on VIM-WN and prove that a multilayer perceptron which has successfully finished learning by VIM-MN is tolerant to all multiple neuron-and-weight faults whose values are in the interval, under the condition that the multiplicity of the multiple fault is within a certain number specified by faulty neurons and weights. Next, we present them concerning VIM-N likewise. By simulation, we confirm the analytical results for VIM-WN and VIM-N. We also by simulation examine the degrees of fault tolerance concerning multiple neuron-and-weight faults for VIM-N and VIM-W where VIM-W is the method proposed in [1] and show that VIM-N and WIM-W as well as VIM-WN are almost equally effective in coping with multiple neuron-and-weight faults. In addition, we show the data in terms of the learning time, successful rate of learning.

Keywords: fault-tolerance, multilayer perceptron, value injection, multiple fault, weight and neuron fault, learning method.

1 Introduction

One of the attractive features of artificial neural networks is their capability to adapt themselves to special environment conditions, by "training" their connection strengths (weights). Especially, feed-forward neural networks with neurons arranged in layers, called the multilayer perceptrons, are widely used in computational or industrial fields. Furthermore, as VLSI technology has developed, the interest in implementing them in hardware is growing. In this case, there is the possibility of low yield and/or reliability of the system, if there is no strategy for coping with defects or faults.

On the other hand, it may be thought that a multilayer perceptron, which is proposed as a model for the cerebral neural network, has a potential ability of fault-tolerance. On this point, Phatak and Koren discussed the fault-tolerance through replications [2]. By the benchmark test for the Sonar nets [3], they showed that more than 99% of all possible single weight faults, which are stuck at $+W$, 0, or $-W$, are tolerated without any additional redundancy, but complete (100%) fault-tolerance is not achieved even at 6 extra replications. Furthermore, Nijhuis et al. showed that fault-tolerance behavior is not self-evident, but it must be activated by an appropriate learning scheme [4]. Since then, many ideas to make the multilayer perceptron fault-tolerant have been studied in the literature, e.g., (see [5] – [13]). These works evaluated their fault-tolerances to the faults with the fixed values like stuck-at faults. On the other hand, one of the authors proposed a value injection method which makes multilayer perceptrons fault-tolerant to multiple weight faults [1]. In addition, some of the authors proposed the "deep learning methods" which makes multilayer perceptrons fault-tolerant to multiple faults of weights and neurons [14]. This is not a value injection method but the extended back-propagation algorithm which adjusts the learning parameters according to the degree of fault-tolerance.

Concerning value injection methods, the works of e.g., [15] – [19] deal with the value injection methods for fault-tolerances of multilayer perceptrons or RBF networks. These works show the convergences and/or effective objective functions of their method, and/or the relation between their injections and output errors such as mean-square errors. Some of the works show them theoretically, but the works does not deal with the methods to obtain 100% fault-tolerance to some ranges of faults.

In this paper, we discuss a fault-tolerance of multilayer perceptrons in which input and output learning examples are patterns consisting of 0s and 1s. A type of faults to be dealt with is a multiple neuron and/or weight fault where neurons are in the hidden layer and weights are between the hidden and output layers. We theoretically analyze the condition that a multilayer perceptron is tolerant to multiple neuron and weight faults. According to the analysis, we propose two value injection methods denoted as VIM-WN and VIM-N to make multilayer perceptrons tolerant to all multiple neuron and/or weight faults whose values are in a multi-dimensional interval. In VIM-WN, the extreme values specified by the fault ranges are set to the outputs of the selected neurons and the selected weights of the links at the same time in a learning phase. In VIM-N, the extreme values specified by the fault ranges are set to the outputs of the selected neurons likewise. First, we present an algorithm based on VIM-WN and prove that a multilayer perceptron which has successfully finished learning by VIM-MN is tolerant to all multiple neuron-and-weight faults whose values are in the interval, under the condition that the multiplicity of the fault is within a certain number specified by faulty neurons and weights. Next, we present an algorithm based on VIM-N and mention without the detail analysis that a multilayer perceptron which has successfully finished learning by VIM-N is tolerant to all multiple neuron faults in a multi-dimensional interval , under the condition that the multiplicity of the fault is within the number of faulty neurons. By simulation, we confirm the analytical results for VIM-WN and VIM-N. We also by simulation examine the degrees of fault tolerance concerning multiple neuron-and-weight faults (simultaneous neuron and weight faults) for VIM-N and VIM-W where VIM-W is the method proposed in [1] and show the interesting result that VIM-N and WIM-W as well as VIM-WN are almost equally effective in coping with multiple neuron-and-weight faults. In addition, we show the data in terms of the learning time, successful rate of learning.

This paper is the extension of [20].

2 Multilayer Perceptron

Fig.1 shows a multilayer perceptron (simply denoted as an "MLP" in the following). Each neuron in a layer is connected to all neurons in the adjacent layers through uni-directional links (synaptic weights). The first and the last layers are called the input and output layers respectively, and one between them is called a hidden layer. In this paper, we deal with only MLPs which have one hidden layer. The output of each neuron (o_i) is given by

$$o_i = f\left(X_i\right) \tag{1}$$

$$X_i = \sum_{j=0}^{N_{pre}} w_{ij} \cdot u_j \tag{2}$$

where w_{ij} is the value of the synaptic weight from the j-th neuron in the preceding layer to the i-th neuron ((i, j) is called to be the index of the weight), N_{pre} is the number of the neurons in the preceding layer connected to the i-th

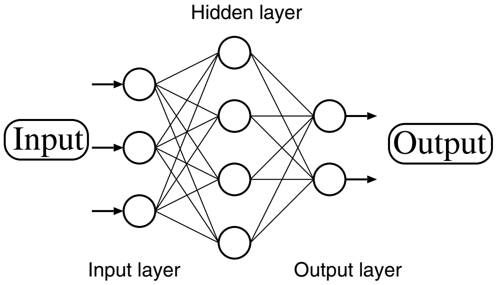


Fig. 1. 3-layer multilayer perceptron

neuron, u_j is the output of the j-th neuron in the preceding layer (j is called to be the index of the neuron), w_{i0} is the synaptic weight connected to the input $u_0 = 1$ corresponding to the threshold, X_i is called the "inner potential" of the i-th neuron, and f is the activation function (the sigmoid function) of a neuron defined by

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{3}$$

The learning process called "back-propagation algorithm" is based on a steepest-descendant gradient rule. Let O be a set of indices of the neurons in the output layer, and let P be a set of indices of the learning input examples. The change of each weight for the p-th learning input example (named w_{ij}^p) is done as follows:

$$\Delta w_{ij}^p = -\eta \cdot \frac{\partial E_p}{\partial w_{ij}} \tag{4}$$

where $E_p = \sum_{i \in O}(t_i^p - o_i^p)^2/2$, $t_i^p(= 0 \text{ or } 1)$ is the learning output example of the i-th neuron in the output layer for the p-th learning input example ($i \in O$ and $p \in P$), o_i^p is the output of the i-th neuron in the output layer for the p-th learning input example, and η is a parameter of a positive real number.

Then, the weight modification is repeated until the following condition is satisfied.

$$\max_{p \in P, i \in O}(t_i^p - o_i^p)^2 < e_o^2 \tag{5}$$

where e_o is called the output error in learning phase. If an MLP obtained by a learning with P and e_o satisfies this condition, the learning is said to have finished successfully and the MLP obtained is called to be "successful" in learning phase in terms of P and e_o (or "successful" in short if no confusion occurs).

3 Fault Model

The three important assumptions concerning faults are denoted as follows.

Assumption 1 *(The range of faults)*
Only neurons in the hidden layer and any weights may be faulty, and other parts (that is, neurons in the input and output layers) are fault-free.

Assumption 2 *(The value of a weight)*
The value of each weight is assumed to be finite and for convenience of analysis, in the range from -1 to 1 even when it is faulty.

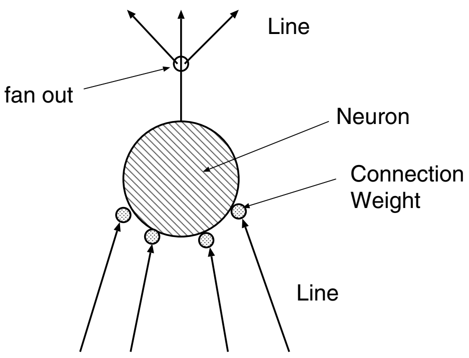


Fig. 2. Elements constructing a neuron

Assumption 3 *(The output value of a neuron)*
The output value of each neuron is assumed to be in the range of the output value of the neuron defined by it's activation function, that is, in the range from 0 *to* 1 *even when it is faulty.*

In the following, we mention that these assumptions are reasonable.

1. Concerning Assumption 1, usually, it is assumed that faults occur at neurons themselves, weights, and interconnecting links (see Fig.2). It can be considered as a weight or a neuron fault that a link snaps or is stuck to some value. It is natural to assume that neurons in the input layer are fault-free because they are only input terminals, that is, so simple circuits. Next, faults of neurons in the output layer are fatal indeed. However, this paper deals with the case that they are fault-free, from the reason that making each neuron in the output layer stronger (that is, more fault-tolerant) than one in the hidden layer, at the fabrication time, is a practical choice, as the number of neurons in the output layer is small.
2. Concerning Assumptions 2 and 3, we deal with the case where values of weights and output values of neurons are finite. It is considered that the case is reasonable in a real world or hardware realization. Then the values of weights are in a range specified by two finite values $+W$ and $-W$ $(W > 0)$. Similarly, output values of neurons are considered to be in a range specified by 0 and U $(U > 0)$. Let the value of a weight be w $(-W \leq w \leq +W)$ and let it's normalized value, that is, divided by W be $\hat{w}$. Then, $w = \hat{w}W$ and $-1 \leq \hat{w} \leq +1$. Similarly, let the output value of a neuron be u and let it's normalized value by U be $\hat{u}$. Then, $u = \hat{u}U$ and $0 \leq \hat{u} \leq 1$. Let the inner potential of a neuron be $X = \sum w_i u_i$. Then, $X/(WU) = \sum \hat{w}_i \hat{u}_i$ and $X/(WU)$ is the normalized inner potential of X. This leads to Eq.(7). Then, it will easily be seen that the general finite case can be analyzed using the normalized inner potential as in the analysis in the next section and the similar results to those in the next section are obtained.

Definition 1. *(Multiple fault) A weight fault is a pair (i, x) of i and x, where i and x denote the index and the value of a faulty weight, and they are said to be the index and the value of the weight fault, respectively. A neuron fault is a pair $[j, y]$ of j and y, where j and y denote the index and the output value of a faulty neuron, and they are said to be the index and the value of the neuron fault, respectively. A successful MLP is said to have a weight fault (i, x) if the value of the weight with index i is stuck to x. Similarly, a successful MLP is said to have a neuron fault $[j, y]$ if the output of the neuron with index j is stuck to y. A set of faults F is called a multiple fault if all the faults in F occur simultaneously. Let $\hat{N}_F = \{j \mid [j, y] \in F\}$ and let $\hat{W}_F = \{i \mid (i, x) \in F\}$. Then, $(\hat{N}_F, \hat{W}_F)$ is called the index set of F.* □

The concepts of fault-tolerance in an MLP are defined as follows.

Definition 2. *(Fault tolerance) Let $o_i^p(F)$ be the output of the i-th neuron in the output layer for the p-th learning input example when a fault F occurs in a successful MLP. If the following inequality is satisfied*

$$\max_{p \in P, i \in O} (t_i^p - o_i^p(F))^2 < e_o^2 \tag{6}$$

the MLP is called to be fault-tolerant to F within the output error of e_o (or simply "fault-tolerant to F" if no confusion occurs). F which does not satisfy Eq.(6) is called to be dangerous. □

Notation:

- I_H denotes a set of indices of neurons in the hidden layer.
- $\hat{N}_F (\subseteq I_H)$ denotes a set of indices of faulty neurons in the hidden layer.
- $\hat{W}_F^i (\subseteq O \times I_H)$ denotes a set of indices of faulty weights from the neurons in the hidden layer to the i-the neuron in the output layer.
- $h(\hat{W}_F^i) = \{j \in I_H | (i, j) \in \hat{W}_F^i\}$.
- $\Gamma((\hat{N}_F, \cup_{i \in O} \hat{W}_F^i) : k)$ denotes a set of all multiple faults with index set $(\hat{N}_F, \cup_{i \in O} \hat{W}_F^i)$ such that $|\hat{N}_F \cup h(\hat{W}_F^i)| \leq k$ for all $i \in O$.

Definition 3. *(Degree of fault tolerance)*

1. *An MLP is called to be k-WNFT if it is fault-tolerant to any multiple fault F in $\Gamma((\hat{N}_F, \cup_{i \in O} \hat{W}_F^i) : k)$. If an MLP is k-WNFT but not $(k+1)$-WNFT, the "degree of multiple WNFT" of the MLP is called to be k.*
2. *An MLP is called to be k-NFT if it is fault-tolerant to any multiple fault F in $\Gamma((\hat{N}_F, \phi) : k)$. If an MLP is k-NFT but not $(k+1)$-NFT, the "degree of multiple NFT" of the MLP is called to be k.*
3. *An MLP is called to be k-WFT if it is fault-tolerant to any multiple fault F in $\Gamma((\phi, \cup_{i \in O} \hat{W}_F^i) : k)$. If an MLP is k-NFT but not $(k+1)$-NFT, the "degree of multiple WFT" of the MLP is called to be k.* □

From the definition, we have the following.

Property 1. If an MLP is k-WNFT, it is k-NFT and k-WFT. □

Now, we discuss the tolerance of an MLP to multiple faults. For convenience of explanation, we suppose that an MLP has only one neuron in the output layer which is denoted as o_{neuron}. Let the p-th learning input example be input to the MLP, and let X^p be the inner potential of o_{neuron}. Then,

$$X^p = \sum_{i \in I_H} w_i u_i \tag{7}$$

where u_i $(1 \geq u_i \geq 0)$ is the output value of the neuron with index i $(i \in I_H)$, and w_i $(1 \geq w_i \geq -1,\ i \in I_H)$ is the weight of the link from the neuron with index i to o_{neuron}.

Let $X^{(p,+,i)}$ be the value of X^p in the MLP when u_i and w_i simultaneously set to $+1$, and let $X^{(p,-,i)}$ be the value of X^p when u_i is set to $+1$ and w_i is set to -1 simultaneously. Let

$$\Delta^{(p,+,i)} = X^{(p,+,i)} - X^p = 1 - w_i u_i \tag{8}$$

and

$$\Delta^{(p,-,i)} = X^{(p,-,i)} - X^p = -1 - w_i u_i \tag{9}$$

Then, $(1 - w_i u_i) \geq 0$ and $(-1 - w_i u_i) \leq 0$ because of Assumptions 2 and 3. We sort $\Delta^{(p,+,i)}$'s $(i \in I_H)$ and $-\Delta^{(p,-,i)}$'s $(i \in I_H)$ in descending order, respectively, as follows.

$$\Delta^{(p,+,q_1)} \geq \Delta^{(p,+,q_2)} \geq \cdots \geq 0,$$

and

$$-\Delta^{(p,-,r_1)} \geq -\Delta^{(p,-,r_2)} \geq \cdots \geq 0$$

We denote the sets of the third elements $*$'s of the superscripts of M largest $\Delta^{(p,+,*)}$'s and $-\Delta^{(p,-,*)}$'s as

$$Q^p_M = \{q_1, ..., q_M\}, \tag{10}$$

and

$$R^p_M = \{r_1, ..., r_M\}, \tag{11}$$

respectively.

Let $X^{(+,Q^p_M)}$ be the inner potential of o_{neuron} when for all i's $\in Q^p_M$ u_i's and w_i's are simultaneously set to 1's, and let $X^{(-,R^p_M)}$ be the inner potential of o_{neuron} when for all i's $\in R^p_M$ u_i's are set to 1's and w_i's to -1's simultaneously. Then,

$$X^{(+,Q^p_M)} = \sum_{i \in Q^p_M} 1 + \sum_{i \in I_H - Q^p_M} w_i u_i \tag{12}$$

$$X^{(-,R^p_M)} = \sum_{i \in R^p_M} -1 + \sum_{i \in I_H - R^p_M} w_i u_i \tag{13}$$

Let X_F^p be the inner potential of o_{neuron} when the MLP has a multiple fault F whose index set is $(\hat{N}_F, \hat{W}_F)$. Then, X_F^p is expressed as

$$\begin{aligned} X_F^p = & \sum_{i \in (I_H - \hat{W}_F - \hat{N}_F)} w_i u_i + \sum_{i \in (\hat{W}_F - \hat{N}_F)} w_i' u_i \\ & + \sum_{i \in (\hat{N}_F - \hat{W}_F)} w_i u_i' + \sum_{i \in \hat{W}_F \cap \hat{N}_F} w_i' u_i' \end{aligned} \tag{14}$$

where w_i' is the value of the faulty weight in F with index i between the hidden and output layers and u_i' is the output value of the faulty neuron in F with index i in the hidden layer.

We have the following results.

Theorem 1. $X^{(+,Q_M^p)} \geq X_F^p \geq X^{(-,R_M^p)}$ for any fault F such that $|\hat{N}_F \cup \hat{W}_F| \leq M$.

The proof is detailed in Appendix A. □

From Theorem 1 and that the activation function is monotonically increasing, we have the following.

Lemma 1. For any fault F such that $|\hat{N}_F \cup \hat{W}_F| \leq M$,

$$f(X^{(+,Q_M^p)}) \geq f(X_F^p) \geq f(X^{(-,R_M^p)}) \tag{15}$$

$$(t^p - f(X^{(+,Q_M^p)}))^2 \geq (t^p - f(X_F^p))^2 \text{ if } t^p = 0. \tag{16}$$

$$(t^p - f(X^{(-,R_M^p)}))^2 \geq (t^p - f(X_F^p))^2 \text{ if } t^p = 1. \tag{17}$$

where f is the activation function defined by Eq.(3) and t^p $(= 0$ or $1)$ is the learning output example of o_{neuron} for the p-th learning input example. □

In a general case where an MLP has n neurons in the output layer, the inner potential of the i-th neuron in the output layer X_i^p for the p-th learning example is denoted as

$X_i^p = \sum_{j \in I_H} w_{ij} u_j$.

We express X_i^p in a convenient form to apply the foregoing analysis to X_i^p as follows.

$X_i^p = \sum_{j \in I_H} w_j^{(i)} u_j$

where $w_j^{(i)} = w_{ij}$. Considering the i-th neuron in the output layer as only one neuron in the output layer, we apply the foregoing analysis to X_i^p expressed in the above form. Then, those corresponding to Q_M^p and R_M^p in Eq.s (10) and (11) are derived , which are denoted as $Q_{i,M}^p$ and $R_{i,M}^p$, respectively. Further, $X_i^{(+,Q_{i,M}^p)}$, $X_i^{(-,R_{i,M}^p)}$ and $X_{i,F}^p$ for the inner potential of i-th neuron, are similarly defined as in Eq.s (12), (13) and (14), respectively. Then, we have the following Lemmas.

Lemma 2. For any fault F with index set $(\hat{N}_F, \hat{W}_F^i)$ such that $|\hat{N}_F \cup h(\hat{W}_F^i)| \leq M$,

$$f(X_i^{(+,Q_{i,M}^p)}) \geq f(X_{i,F}^p) \geq f(X_i^{(-,R_{i,M}^p)}) \tag{18}$$

$$(t_i^p - f(X_i^{(+,Q_{i.M}^p)}))^2 \geq (t_i^p - f(X_{i,F}^p))^2 \text{ if } t_i^p = 0. \tag{19}$$

$$(t_i^p - f(X_i^{(-,R_{i,M}^p)}))^2 \geq (t_i^p - f(X_{i,F}^p))^2 \text{ if } t_i^p = 1. \tag{20}$$

□

Lemma 3. Let an MLP have n neurons in the output layer. Let F be any multiple fault in $\Gamma((\hat{N}_F, \cup_{i\in O}\hat{W}_F^i) : M)$. Then, for $i \in O(1 \leq i \leq n)$ and $p \in P$ such that $t_i^p = 0$,

$$\max_{p\in P, 1\leq k\leq n, t_k^p=0} (t_k^p - f(X_k^{(+,Q_{k,M}^p)}))^2 \geq (t_i^p - f(X_{i,F}^p))^2, \tag{21}$$

and for $i \in O(1 \leq i \leq n)$ and $p \in P$ such that $t_i^p = 1$,

$$\max_{p\in P, 1\leq k\leq n, t_k^p=1} (t_k^p - f(X_k^{(-,R_{k,M}^p)}))^2 \geq (t_i^p - f(X_{i,F}^p))^2. \tag{22}$$

□

Theorem 2. *An MLP with n neurons in the output layer is fault-tolerant to any multiple fault F in $\Gamma((\hat{N}_F, \cup_{i\in O}\hat{W}_F^i) : M)$ if and only if*

$$\max_{p\in P, 1\leq k\leq n, t_k^p=0} (t_k^p - f(X_k^{(+,Q_{k,M}^p)}))^2 < e_o^2 \tag{23}$$

and

$$\max_{p\in P, 1\leq k\leq n, t_k^p=1} (t_k^p - f(X_k^{(-,R_{k,M}^p)}))^2 < e_o^2 \tag{24}$$

Proof. The if-part: Suppose that Eq.s (23) and (24) hold and there is a dangerous multiple fault F in $\Gamma((\hat{N}_F, \cup_{i\in O}\hat{W}_F^i) : M)$. Then, $(t_i^p - X_{i,F}^p)^2 \geq e_o^2$ for some $p \in P$ and $i \in O$ where $X_{i,F}^p = o_i^p(F)$. From Lemma 3, this contradicts Eq. (23) if $t_p^i = 0$ and Eq. (24) if $t_p^i = 1$.

The only-if part: Suppose that An MLP is fault-tolerant to any multiple fault F in $\Gamma((\hat{N}_F, \cup_{i\in O}\hat{W}_F^i) : M)$ and at least one of Eq.s (23) and (24) does not hold. Suppose that Eq. (23) does not hold. Then, $(t_k^p - f(X_k^{(+,Q_{k,M}^p)}))^2 \geq e_o^2$ for some $p \in P$, some $k \in O$ and $t_k^p = 0$. Let $F = \{[j, +1], ((k, j), +1)|j \in Q_{k,M}^p\}$. Then, since $f(X_k^{(+,Q_{k,M}^p)}) = o_k^p(F)$, F is a dangerous fault. On the other hand, the index set $(\hat{N}_F, \cup_{i\in O}\hat{W}_F^i)$ of F is as follows: $\hat{N}_F = \{j|j \in Q_{k,M}^p\} = Q_{k,M}^p$, $W_F^k = \{(k, j)|j \in Q_{k,M}^p\}$ and $W_F^i = \phi$ for $i \in O \neq k$. Hence, since $|\hat{N}_F \cup h(W_F^k)| = |Q_{k,M}^p| = M$ and $|\hat{N}_F \cup h(W_F^i)| = |\hat{N}_F| = M$ for $i \in O \neq k$, F is in $\Gamma((\hat{N}_F, \cup_{i\in O}\hat{W}_F^i) : M)$. This is a contradiction. For the case where Eq. (24) does not hold, a contradiction is led similarly. □

Property 2. An MLP is M-WNFT if and only if $\max_{p\in P, 1\le k\le n, t_k^p=0}(t_k^p - f(X_k^{(+,Q_{k,M}^p)}))^2 < e_o^2$ *and* $\max_{p\in P, 1\le k\le n, t_k^p=1}(t_k^p - f(X_k^{(-,R_{k,M}^p)}))^2 < e_o^2$. □

Based on the above analysis, we present methods to make MLPs tolerant to multiple neuron and/or weight faults.

4 Value Injection Methods

The value injection methods set the extreme values specified by the fault ranges to the outputs of the selected neurons and/or the selected weights of the links in an MLP at the same time in a learning phase, and applies a back-propagation algorithm to the MLP. Concerning how the values are injected into outputs of neurons and/or weights, the three methods can be considered, that is, injecting into only weights (of links), only (outputs of) neurons, and both neurons and weights. The first method denoted here as VIM-W has already been proposed in [1]. The second method denoted as VIM-N and the third one denoted as VIM-WN are presented here. A learning algorithm for VIM-WN is given, based on the analytical processes in the preceding section. Then, from Theorem 2, it is seen that an MLP which has successfully finished learning by the algorithm is tolerant to all multiple neuron-and-weight faults whose values are in a multi-dimensional interval defined by the extreme values that neurons and weights can take, under the condition that the multiplicity of multiple faults is within the number M. The similar result for VIM-N is expressed though the theoretical analysis is omitted because VIM-N is similar to VIM-WN but the injection is done to only neurons.

4.1 VIM-WN

(1) Overview

- Value injection
  The values 1's are injected into neurons in the hidden layer and 1's or -1's into weights between the hidden and output layers simultaneously in each learning phase, where the indices of neurons and weights to be injected into are given as in Eq.s (10) and (11). The numbers of elements in these sets are the same with each other in a learning phase and denoted as "M_I".
- Weight modification
  Just before an injection, the values of weights in the set are temporarily stored in the variables. After the injection, it is checked whether Eq.(5) is satisfied. If it is satisfied, the learning process successfully ends. Otherwise, the weight modification is executed, the values of the weights in the set are restored using the values of the variables, and the process above is repeated.

(2) Algorithm
The algorithm is constructed according to the flow of the analysis from Eq.s (7) to (21).

[VIM-WN algorithm]

1. Set a positive integer to M_I.
2. Initialize the value of each weight by a pseudo-random floating point number from -0.1 to 0.1.
3. For each $p \in P$, evaluate the output value of each neuron in the hidden layer according to Eq. (1), and for each $i \in O$, evaluate the inner potential X_i^p of the i-th neuron in the output layer according to Eq. (2).
4. For each (p, i) where $p \in P$ and $i \in O$, do the following.
   (a) For each $j \in I_H$,
   If $t_i^p = 0$, evaluate $(1 - w_{ij} \cdot u_j)$.
   if $t_i^p = 1$, evaluate $-(-1 - w_{ij} \cdot u_j)$.
   Let the value evaluated be Δ_{ij}^p.
   (b) Let $S_{M_I}^{(p,i)}$ be the set of the indices j's of M_I largest Δ_{ij}^ps.
5. For each (p, i) where $p \in P$ and $i \in O$, do the following.
   if $t_i^p = 0$, evaluate $-\left(X_i^p + \sum_{j \in S_{M_I}^{(p,i)}} \Delta_{ij}^p\right)$.
   if $t_i^p = 1$, evaluate $\left(X_i^p + \sum_{j \in S_{M_I}^{(p,i)}} (-\Delta_{ij}^p)\right)$.
   Let the value evaluated be $X^{(p,i)}$ and let (p_E, i_E) be the superscript of $X^{(p,i)}$ which is the smallest among all $X^{(p,i)}$s for $p \in P$ and $i \in O$, .
6. Set the output value of each neuron in the hidden layer whose index is in $S_{M_I}^{(p_E,i_E)}$ to 1.
7. Temporarily store the value of each weight $w_{i_E j}$ ($j \in S_{M_I}^{(p_E,i_E)}$) to a variable var_j, and set it to 1 if $t_{i_E}^{p_E} = 0$ (to -1 if $t_{i_E}^{p_E} = 1$).
8. Evaluate the output values of all the neurons in the output layer for each $p \in P$. If Eq.(5) is satisfied, the algorithm successfully ends, and otherwise, go to the next step.
9. Modify the values of all the weights according to Eq.(4).
10. Restore the value in the variable var_j in Step 7 to the value of the weight $w_{i_E,\, j}$ for each $j \in S_{M_I}^{(p_E,i_E)}$.
11. Change the value of each weight to 1 (-1) if it is greater (less) than 1 (-1). This process is called "$W_{|1|}$-process". Note that this process is done for making each weight within 1 to -1.
12. Go to Step 3. □

Notation:

– An MLP which has successfully finished by VIM-WN algorithm with M_I is denoted as an "MLP-WN(M_I)".

The correspondence between the algorithm and the flow of the analysis from Eq.s (7) to (20) is as follows.

1. Step 4(a) corresponds to Eq. (8) or (9). Step 4(b) corresponds to Eq.(10) or (11).

2. Step 5 corresponds to the negative value of Eq. (12) for $i \in O$ $-X_i^{(+,Q^p_{i,M})}$ if $t^p_i = 0$ and the value of Eq.(13) for $i \in O$ $X_i^{(-,R^p_{i,M})}$ if $t^p_i = 1$. p_E and i_E correspond to the p and k which realize the maximum of the left side in Eq. (19) or (20). Note that for an activation function f, $f(x) = 1 - f(-x)$.
3. At Steps 6 and 7, the values are injected into the outputs of neurons in the hidden layer and the weights between the hidden and output layers, according to $S^{(p_E,i_E)}_{M_I}$. At Step 8, the left side of Eq.(19) or (20) is evaluated. The values of the weights before the values are injected into them are temporarily stored in the variables *var*'s, but not so are the values of the neurons. The values in the *var*'s at Step 10 are restored to the former, and the latter are restored by recalculation at the next modification timing.
4. Step 11 corresponds to Assumptions 2 and 3.

We have the following property from the correspondence between the algorithm and the flow of the analysis mentioned above.

Property 3. An MLP-WN(M_I) is M_I-WNFT. □

4.2 VIM-N

VIM-N is similar to VIM-WN but the injection is done to only neurons. However, the method will be presented because as shown in the simulation, it is so useful to realize fault-tolerance to simultaneous neuron and weight faults as well as only neuron faults.

[VIM-N algorithm]

```
1. Set a positive integer to $M_I$.
2. Initialize the value of each weight by a pseudo-random floating point number from $-0.1$ to $0.1$.
3. Evaluate the output value of each neuron in the hidden and output layers and the inner potential $X^p_i$ for each $(p,i)$ like in VIM-WN.
4. For each $(p, i)$ where $p \in P$ and $i \in O$, do the following.
   (a) If $t^p_i = 0$, for each $j \in I_H$, evaluate $(w_{ij} \cdot 1 - w_{ij}u_j)$ if $w_{ij} > 0$, and $(w_{ij} \cdot 0 - w_{ij}u_j)$ otherwise. Let the value evaluated be $\Delta^p_{ij}$.
   (b) If $t^p_i = 1$, for each $j \in I_H$, evaluate $-(w_{ij} \cdot 0 - w_{ij}u_j)$ if $w_{ij} > 0$, and $-(w_{ij} \cdot 1 - w_{ij}u_j)$ otherwise. Let the value evaluated be $\Delta^p_{ij}$.
   (c) Let $S^{(p,i)}_{M_I}$ be the set of the indices $j$'s of $M_I$ largest $\Delta^p_{ij}$s.
5. For each $(p, i)$ where $p \in P$ and $i \in O$, do the following.
   if $t^p_i = 0$, evaluate $-\ (X^p_i + \sum_{j \in S^{(p,i)}_{M_I}} \Delta^p_{ij})$.
   if $t^p_i = 1$, evaluate $(X^p_i + \sum_{j \in S^{(p,i)}_{M_I}} (-\Delta^p_{ij}))$.
   Let the value evaluated be $X^{(p,i)}$ and let $(p_E, i_E)$ be the superscript of $X^{(p,i)}$ which is the smallest among all $X^{(p,i)}$s for $p \in P$ and $i \in O$,
6. Set the output value of each neuron in the hidden layer whose index is in $S^{(p_E,i_E)}_{M_I}$ to 1 if $(t^{p_E}_{i_E} = 0$ and $w_{i_E j} > 0)$ or $(t^{p_E}_{i_E} = 1$ and $w_{i_E j} \leq 0)$, and 0 otherwise.
```

7. Evaluate the output values of all the neurons in the output layer for each $p \in P$. If Eq.(5) is satisfied, the algorithm successfully ends, and otherwise, go to the next step.
8. Modify the values of all the weights according to Eq.(4).
9. Do the $W_{|1|}$-process.
10. Go to Step 3. □

Notation:

– An MLP obtained by a successful VIM-N learning with M_I is denoted as an MLP-N(M_I).

We has the following property.

Property 4. An MLP-N(M_I) is M_I-NFT.
The proof is omitted because it could be done like in deriving Theorem 2 and Property 2. □

We briefly introduce VIM-W which was presented in [1] to see the ability of tolerance to multiple neuron-and-weight faults. This method injects the extreme values +1 (−1)s into the weights selected under the certain criterion in a learning phase (See [1]). An MLP obtained by a successful VIM-W learning with M_I is denoted as an MLP-W(M_I) where M_I is the number of the selected weights as above. Then, the following has been proved [1].

Property 5. An MLP-W(M_I) is M_I-WFT. □

5 Simulation Results

We perform a simulation of a character recognition to see the convergence of learning and confirm the analytical results on VIM-WN and VIM-N. We show the learning times, the rates of successful learning, and the degrees of multiple fault tolerance to neuron and/or weight faults for VIM-WN, -N and -W.

The simulations are done using a PC with an Athlon X2 3800+, 512MB RAM, and the Fedora Core 6. Simulation programs are described in C language, and the gcc version 4.1.1 compiler is used.

Concerning the parameters, the maximum weight update is 5×10^5, $e_o = e_u = 0.1$, and $\eta = 0.1$.

Fig. 3 shows the set of totally 20 learning input examples. Each example consists of 100 bit signals (black=1 and white=0). Table 1 shows the learning output examples. The number of neurons in the input layer is set to 100. The number of neurons in the output layer is set to the minimum one according to the number of learning input examples to be used.

The case that a simulation is performed to an MLP with N_h neurons in the hidden layer and N_{le} learning input examples is denoted as "N_h_N_{le}", The simulation has been performed for several cases. Here, the results for the two cases of 100_10 and 50_20 are shown.

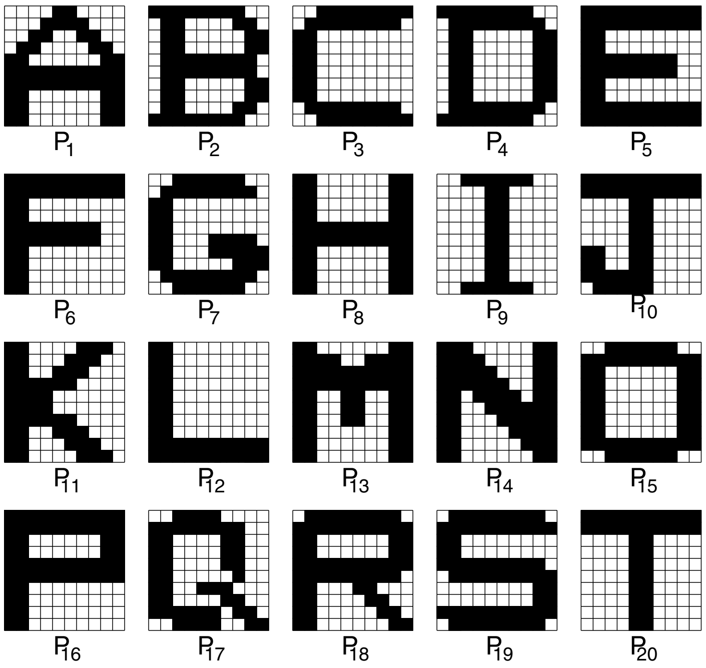


Fig. 3. Learning input examples (*Set*-1)

For each case, to get the maximum M_I, first, M_I is set to 0. Then, the execution of the algorithm of VIM-WN, -N or -W begins. The algorithm is executed 100 times until the algorithm successfully ends. If the algorithm successfully ends, M_I is increased by 1 and the algorithm is repeated from the beginning. Otherwise, the execution of the algorithm is finished. In this way, The maximum M_I is attained for each case in each method., which is denoted as $\hat{M}_I$.

(1) Rate of successful learning and learning time

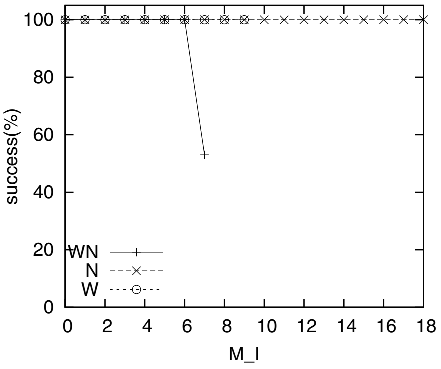
Figs.4 and 5 show the rates of successful learnings (denoted as "success") and the learning times (denoted as "time") for M_I for two cases of 100_10 and 50_20, respectively. They are the average values for 100 learning trials. The labels "WN", "N", and "W" indicate the methods of VIM-WN, -N and -W, respectively.

From these data, The $\hat{M}_I$s of the three methods for the cases of 100_10 and 50_20 are as shown in Tables 2 . Further, the following are seen.

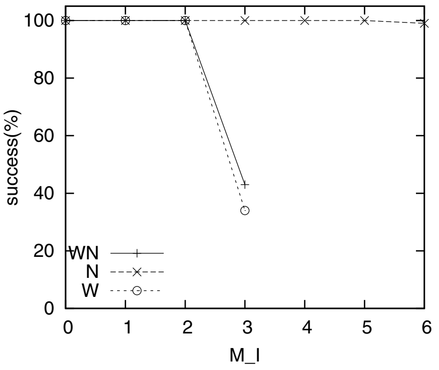
1. The rates of successful learnings become less and the learning times become bigger as the values of M_I become bigger.
2. $\hat{M}_I$s become less as the ratios N_h/N_{le} of N_h and N_{le} become less.
3. The learning times in the three methods when $\hat{M}_I$s have been attained, are almost the same with each other.

Table 1. Learning output examples

input	output $t_4^p t_3^p t_2^p t_1^p t_0^p$
P_1	00000
P_2	00001
P_3	00010
omitted	
P_{24}	10111
P_{25}	11000



(a) 100_10

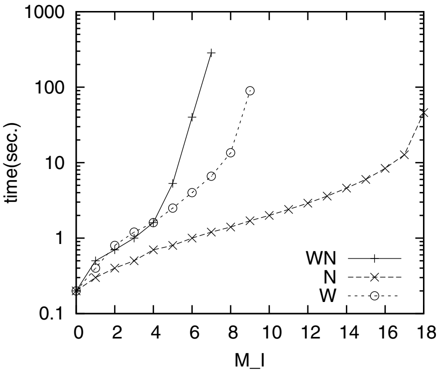


(b) 50_20

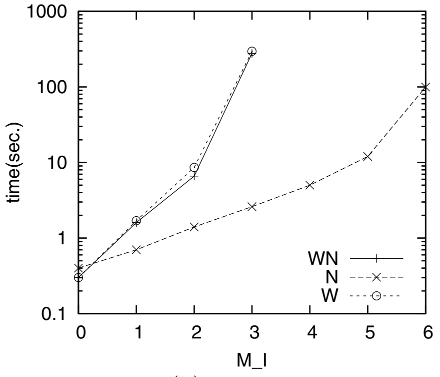
Fig. 4. The relation between rate of successful learnings and M_I

Table 2. The maximum M_Is of VIM-WN(M_I), -N(M_I) and -W(M_I)

	VIM-N	VIM-WN	VIM-W
100_10	18	7	9
50_20	6	3	3



(a) 100_10



(b) 50_20

Fig. 5. The relation between learning time and M_I

Table 3. Degree of multiple WNFT

	MLP-N	MLP-WN	MLP-W
100_10	8	7	8
50_20	2	3	2

Table 4. Degree of multiple NFT

	MLP-N	MLP-WN	MLP-W
100_10	18	13	15
50_20	6	5	5

(2) Degree of multiple fault tolerance

We examine the degrees of multiple WNFT and NFT of MLP-WN($\hat{M}_I$)s, MLP-N($\hat{M}_I$)s and MLP-W($\hat{M}_I$)s, by checking the condition in Property 2 for a million M-multiple faults per M, which are generated using pseudo-random numbers, increasing M one by one. Tables 3 and 4 show their maximum Ms which satisfy Property 2, that is, the degrees of multiple WNFT and NFT of MLP-WN($\hat{M}_I$)s, MLP-N($\hat{M}_I$)s and MLP-W($\hat{M}_I$)s for the two $N_h\_N_{le}$ cases. From the data of Tables 2, 3 and 4, the following are seen.

1. The degree of multiple WNFT of the MLP-WN(7) is 7, that is, the MLP-WN(7) is 7-WNFT, as Property 3 shows.
2. The degree of multiple NFT of the MLP-N(18) is 18, that is, the MLP-N(18) is 18-NFT, as Property 4 shows.
3. All of the VIM methods are almost equally effective to realize the fault tolerance to simultaneous neuron and weight faults, because the values in Tables 3 and 4 of the MLP-WN($\hat{M}_I$), MLP-N($\hat{M}_I$) and MLP-W($\hat{M}_I$) in each $N_h\_N_{le}$ case are almost the same with each other.

6 Conclusions

We have theoretically analyzed the condition that a multilayer perceptron is tolerant to multiple neuron and weight faults. The condition can be used to check whether an MLP is tolerant to multiple neuron and weight faults. Further, according to the flow of the analysis, we propose two value injection learning methods denoted as VIM-WN and VIM-N to make multilayer perceptrons tolerant to all multiple neuron and/or weight faults whose values are in a multi-dimensional interval defined by the extreme values that neurons and weights can take. A simulation is performed to see the convergence of the learning and confirm the analytical results. The simulation results show that VIM-N and VIM-W as well as VIM-WN are almost equally effective in coping with multiple neuron-and-weight faults. This result is so interesting though the condition that VIM-WN satisfies is not theoretically guaranteed for VIM-N and VIM-W. It is hoped that it will be theoretically analyzed for VIM-N and VIM-W, if possible. This is a future work.

Acknowledgment. The author would like to thank Mr. M. Otsu for prototype works of learning simulations.

Appendix A: Proof of Theorem 1

First, we prove $X^{(p,+,Q_M^p)} \geq X_F^p$ by proving the following lemmas.

Lemma 4. For $i \in Q_M^p$, $j \in (I_H - Q_M^p)$ and any x $(+1 \geq x \geq -1)$, $(1 - w_i u_i)$ $(= \Delta^{(p,+,i)}) \geq (x - w_j)u_j$.

Proof: Suppose $(1 - w_i u_i) < (x - w_j)u_j$. Then, since $0 \leq (1 - x \cdot u_j) = (1 - w_j u_j + w_j u_j - x \cdot u_j)$, $\Delta^{(p,+,i)} = (1 - w_i u_i) < (x - w_j)u_j \leq (1 - w_j u_j) = \Delta^{(p,+,j)}$. Hence, $j \in Q_M^p$ which conflicts with $j \in (I_H - Q_M^p)$. □

Lemma 5. For $i \in Q_M^p$, $j \in (I_H - Q_M^p)$ and any x $(+1 \geq x \geq -1)$, $(1 - w_i u_i)$ $(= \Delta^{(p,+,i)}) \geq w_j(x - u_j)$.

Proof: Suppose $(1 - w_i u_i) < w_j(x - u_j)$. Then, since $0 \leq (1 - w_j \cdot x) = (1 - w_j u_j + w_j u_j - w_j \cdot x)$, $\Delta^{(p,+,i)} = (1 - w_i u_i) < w_j(x - u_j) \leq (1 - w_j u_j) = \Delta^{(p,+,j)}$. Hence, $j \in Q_M^p$ which conflicts with $j \in (I_H - Q_M^p)$. □

Lemma 6. For $i \in Q_M^p$, $j \in (I_H - Q_M^p)$, any x $(+1 \geq x \geq -1)$ and any y $(+1 \geq y \geq 0)$, $(1 - w_i u_i)$ $(= \Delta^{(p,+,i)}) \geq (x \cdot y - w_j u_j)$.

Proof: Suppose $(1 - w_i u_i) < (x \cdot y - w_j u_j)$. Then, since $0 \leq (1 - x \cdot y) = (1 - w_j u_j + w_j u_j - x \cdot y)$, $\Delta^{(p,+,i)} = (1 - w_i u_i) < (x \cdot y - w_j u_j) \leq (1 - w_j u_j) = \Delta^{(p,+,j)}$. Hence, $j \in Q_M^p$ which conflicts with $j \in (I_H - Q_M^p)$. □

Now, we will prove $X^{(p,+,Q_M^p)} \geq X_F^p$ if $|\hat{W}_F \cup \hat{N}_F| \leq M$.

$$\begin{aligned} X^{(p,+,Q_M^p)} - X_F^p = & \sum_{i \in Q_M^p \cap \hat{N}_F \cap \hat{W}_F} (1 - w_i' u_i') \\ & + \sum_{i \in Q_M^p \cap (\hat{W}_F - \hat{N}_F)} (1 - w_i' u_i) \\ & + \sum_{i \in Q_M^p \cap (\hat{N}_F - \hat{W}_F)} (1 - w_i u_i') \\ & + \sum_{i \in (\hat{W}_F - \hat{N}_F - Q_M^p)} (w_i - w_i') u_i \\ & + \sum_{i \in (\hat{N}_F \cap \hat{W}_F - Q_M^p)} (w_i u_i - w_i' u_i') \\ & + \sum_{i \in (\hat{N}_F - \hat{W}_F - Q_M^p)} w_i (u_i - u_i') \\ & + \sum_{i \in (Q_M^p - (\hat{N}_F \cup \hat{W}_F))} (1 - w_i u_i) \end{aligned} \quad (25)$$

Let's denote the first, second, ..., the seventh terms of Eq. (25) as S_1, S_2, ..., S_7, respectively. It is clear that $S_1 \geq 0$, $S_2 \geq 0$, $S_3 \geq 0$ and $S_7 \geq 0$. In the following, we will show $(S_4 + S_5 + S_6 + S_7) \geq 0$.

We have the following lemma.

Lemma 7

$$\begin{aligned} (\hat{W}_F - \hat{N}_F - Q_M^p) \cup (\hat{N}_F \cap \hat{W}_F - Q_M^p) \\ \cup (\hat{N}_F - \hat{W}_F - Q_M^p) = \hat{N}_F \cup \hat{W}_F - Q_M^p \end{aligned} \quad (26)$$

$$\begin{aligned} |(\hat{W}_F - \hat{N}_F - Q_M^p)| + |(\hat{N}_F \cap \hat{W}_F - Q_M^p)| \\ + |(\hat{N}_F - \hat{W}_F - Q_M^p)| \leq |Q_M^p - (\hat{N}_F \cup \hat{W}_F)| \end{aligned} \quad (27)$$

Proof: Eq. (26) can easily be proved. Eq. (27) holds because of Eq. (26) and $|\hat{W}_F \cup \hat{N}_F| \leq M = |Q_M^p|$. □

Without lost of generality, we can put

$$S_7 = (1 - w_{i_1}u_{i_1}) + \cdots + (1 - w_{i_v}u_{i_v}),$$
$$\text{each } i_d \in (Q^p_M - (\hat{N}_F \cap \hat{W}_F))$$

and

$$(1 - w_{i_1}u_{i_1}) \geq \cdots \geq (1 - w_{i_v}u_{i_v}) \geq 0.$$

Suppose $(S_4 + S_5 + S_6 + S_7) < 0$. Note that from Eq.(27) the number of the terms in the summations of S_4, S_5 and S_6 is not more than the number of the terms in the summation of S_7. Then, at least one of the following should hold from Eq. 25 and Lemma 7.

(a) $(w'_j - w_j)u_j > (1 - w_{i_v}u_{i_v})$ for some $j \in (\hat{W}_F - \hat{N}_F - Q^p_M)$.
(b) $(w'_j u'_j - w_j u_j) > (1 - w_{i_v}u_{i_v})$ for some $j \in (\hat{N}_F \cap \hat{W}_F - Q^p_M)$.
(c) $w_i(u'_j - u_j) > (1 - w_{i_v}u_{i_v})$ for some $j \in (\hat{N}_F - \hat{W}_F - Q^p_M)$.

(a) does not hold from Lemma 4 since $i_v \in Q^p_M$ and $j \in (\hat{W}_F - \hat{N}_F - Q^p_M) \subset (I_H - Q^p_M)$. (c) does not hold from Lemma 5 since $i_v \in Q^p_M$ and $j \in (\hat{N}_F - \hat{W}_F - Q^p_M) \subset (I_H - Q^p_M)$. (b) does not hold from Lemma 6 since $i_v \in Q^p_M$ and $j \in (\hat{N}_F \cap \hat{W}_F - Q^p_M) \subset (I_H - Q^p_M)$. Hence, $(S_4 + S_5 + S_6 + S_7) \geq 0$ and $X^{(p,+,Q^p_M)} - X^p_F \geq 0$.

Next, we prove $X^p_F \geq X^{(-,R^p_M)}$ by proving the following lemmas.

Lemma 8. For $i \in R^p_M$, $j \in (I_H - R^p_M)$ and any x $(+1 \geq x \geq -1)$, $(-1 - w_i u_i)$ $(= \Delta^{(p,-,i)}) \leq (x - w_j)u_j$.

Proof: Suppose $(-1 - w_i u_i) > (x - w_j)u_j$. Then, since $0 \geq (-1 - x \cdot u_j) = (-1 - w_j u_j + w_j u_j - x \cdot u_j)$, $\Delta^{(p,-,i)} = (-1 - w_i u_i) > (x - w_j)u_j \geq (-1 - w_j u_j) = \Delta^{(p,-,j)}$. Hence, $-\Delta^{(p,-,i)} < -\Delta^{(p,-,j)}$ and $j \in R^p_M$ which conflicts with $j \in (I_H - R^p_M)$. □

Lemma 9. For $i \in R^p_M$, $j \in (I_H - R^p_M)$ and any x $(+1 \geq x \geq -1)$, $(-1 - w_i u_i)$ $(= \Delta^{(p,-,i)}) \leq w_j(x - u_j)$.

Proof: Suppose $(-1 - w_i u_i) > w_j(x - u_j)$. Then, since $0 \geq (-1 - w_j \cdot x) = (-1 - w_j u_j + w_j u_j - w_j \cdot x)$, $\Delta^{(p,-,i)} = (-1 - w_i u_i) > w_j(x - u_j) \geq (-1 - w_j u_j) = \Delta^{(p,-,j)}$. Hence, $j \in R^p_M$ which conflicts with $j \in (I_H - R^p_M)$. □

Lemma 10. For $i \in R^p_M$, $j \in (I_H - R^p_M)$, any x $(+1 \geq x \geq -1)$ and any y $(+1 \geq y \geq 0)$, $(-1 - w_i u_i)$ $(= \Delta^{(p,-,i)}) \leq (x \cdot y - w_j u_j)$.

Proof: Suppose $(-1 - w_i u_i) > (x \cdot y - w_j u_j)$. Then, since $0 \geq (-1 - x \cdot y)$ $= (-1 - w_j u_j + w_j u_j - x \cdot y)$, $\Delta^{(p,-,i)} = (-1 - w_i u_i) > (x \cdot y - w_j u_j) \geq (-1 - w_j u_j) = \Delta^{(p,-,j)}$. Hence, $j \in R^p_M$ which conflicts with $j \in (I_H - R^p_M)$. □

Now, we will prove $X^{(p,-,R^p_M)} \leq X^p_F$ if $|\hat{W}_F \cup \hat{N}_F| \leq M$.

$$\begin{aligned} X^{(p,-,R_M^p)} - X_F^p = & \sum_{i \in R_M^p \cap \hat{N}_F \cap \hat{W}_F} (-1 - w_i' u_i') \\ & + \sum_{i \in R_M^p \cap (\hat{W}_F - \hat{N}_F)} (-1 - w_i' u_i) \\ & + \sum_{i \in R_M^p \cap (\hat{N}_F - \hat{W}_F)} (-1 - w_i u_i') \\ & + \sum_{i \in (\hat{W}_F - \hat{N}_F - R_M^p)} (w_i - w_i') u_i \\ & + \sum_{i \in (\hat{N}_F \cap \hat{W}_F - R_M^p)} (w_i u_i - w_i' u_i') \\ & + \sum_{i \in (\hat{N}_F - \hat{W}_F - R_M^p)} w_i (u_i - u_i') \\ & + \sum_{i \in (R_M^p - (\hat{N}_F \cup \hat{W}_F))} (-1 - w_i u_i) \end{aligned} \tag{28}$$

Let's denote the first, second, ..., the seventh terms of Eq. (28) as T_1, T_2, ..., T_7, respectively. It is clear that $T_1 \leq 0$, $T_2 \leq 0$, $T_3 \leq 0$ and $T_7 \leq 0$. In the following, we will show $(T_4 + T_5 + T_6 + T_7) \leq 0$.

We have the following lemma.

Lemma 11

$$\begin{aligned} (\hat{W}_F - \hat{N}_F - R_M^p) \cup (\hat{N}_F \cap \hat{W}_F - R_M^p) \\ \cup (\hat{N}_F - \hat{W}_F - R_M^p) = \hat{N}_F \cup \hat{W}_F - R_M^p \end{aligned} \tag{29}$$

$$\begin{aligned} |(\hat{W}_F - \hat{N}_F - R_M^p)| + |(\hat{N}_F \cap \hat{W}_F - R_M^p)| \\ + |(\hat{N}_F - \hat{W}_F - R_M^p)| \leq |R_M^p - (\hat{N}_F \cup \hat{W}_F)| \end{aligned} \tag{30}$$

Proof: Similar to the proof of Lemma 7. □

Without lost of generality, we can put

$$T_7 = (-1 - w_{i_1} u_{i_1}) + \cdots + (-1 - w_{i_v} u_{i_v}), \\ \text{each } i_d \in (R_M^p - (\hat{N}_F \cap \hat{W}_F))$$

and

$$-(-1 - w_{i_1} u_{i_1}) \geq \cdots \geq -(-1 - w_{i_v} u_{i_v}) \geq 0.$$

Suppose $(T_4 + T_5 + T_6 + T_7) > 0$. Note that from Eq. (30) the number of the terms in the summations of T_4, T_5 and T_6 is not more than the number of the terms in the summation of T_7. Then, at least one of the following should hold from Eq. (28) and Lemma 11.

(a) $(w'_j - w_j)u_i < (-1 - w_{i_v}u_{i_v})$ for some $j \in (\hat{W}_F - \hat{N}_F - R^p_M)$.
(b) $(w'_j u'_j - w_j u_j) < (-1 - w_{i_v}u_{i_v})$ for some $j \in (\hat{N}_F \cap \hat{W}_F - R^p_M)$.
(c) $w_j(u'_j - u_j) < (-1 - w_{i_v}u_{i_v})$ for some $j \in (\hat{N}_F - \hat{W}_F - R^p_M)$.

(a) does not hold from Lemma 8 since $i_v \in R^p_M$ and $j \in (\hat{W}_F - \hat{N}_F - R^p_M) \subset (I_H - R^p_M)$. (c) does not hold from Lemma 9 since $i_v \in R^p_M$ and $j \in (\hat{N}_F - \hat{W}_F - R^p_M) \subset (I_H - R^p_M)$. (b) does not hold from Lemma 10 since $i_v \in R^p_M$ and $j \in (\hat{N}_F \cap \hat{W}_F - R^p_M) \subset (I_H - R^p_M)$. Hence, $(T_4 + T_5 + T_6 + T_7) \leq 0$ and $X^{(p,-,R^p_M)} - X^p_F \leq 0$.
From the above, Theorem 1 is proved. □

References

1. Takanami, I., Oyama, Y.: A novel learning algorithm which makes multilayer neural networks multiple-weight-fault tolerant. IEICE Trans. Inf. & Syst. E86-D(12), 2536–2543 (2003)
2. Phatak, D.S., Koren, I.: Complete and partial fault tolerance of feedforward neural nets. IEEE Trans. Neural Networks 6(2), 446–456 (1995)
3. Fahlman, S.E., et al.: Neural nets learning algorithms and benchmarks database. Maintained by S.E. Fahlman et.al. at the Computer Science Dept., Carnegie Mellon University
4. Nijhuis, J., Hoefflinger, B., van Schaik, A., Spaanenburg, L.: Limits to the fault-tolerance of a feedforward neural network with learning. In: Proc. Int'l Symp. on FTCS, pp. 228–235 (1990)
5. Tan, Y., Nanya, T.: A faut-tolerant multi-layer neural network model and its properties. IEICE D-I J76-D-I(7), 380–389 (1993) (in Japanese)
6. Clay, R.D., Séquin, C.H.: Fault tolerance training improves generalization and robustness. In: Proc. Int'l. J. Conf. on Neural Networks, pp. I-769–I-774 (1992)
7. Ito, T., Takanami, I.: On fault injection approaches for fault tolerance of feedforward neural networks. In: Proc. Int'l Symp. on ATS, pp. 88–93 (1997)
8. Hammadi, N.C., Ito, H.: A learning algorithm for fault tolerant feedforward neural networks. IEICE Trans. Inf & Syst. E80-D(1), 21–26 (1997)
9. Hammadi, N.C., Ohmameuda, T., Kaneko, K., Ito, H.: Dynamic constructive fault tolerant algorithm for feedforward neural networks. IEICE Trans. Inf & Syst. E81-D(1), 115–123 (1998)
10. Cavalieri, S., Mirabella, O.: A novel learning algorithm which impoves the partial fault tolerance of multilayer neural networks. Neural Networks (Pergamon) 12(1), 91–106 (1999)
11. Kamiura, N., Hata, Y., Matsui, N.: Fault tolerant feedforward neural networks with learning algorithm based on synaptic weight limit. In: Proc. IEEE Int'l Workshop on On-Line Testing, pp. 222–226 (1999)
12. Kamiura, N., Taniguchi, Y., Hata, Y., Matsui, N.: A learning algorithm with activation function manipulation for fault tolerant neural networks. IEICE Trans. Inf. & Syst. E84-D(7), 899–905 (2001)
13. Takase, H., Kita, H., Hayashi, T.: Weight minimization approach for fault tolerant multi-layer neural networks. In: Proc. of Int'l J. Conf. on Neural Networks, pp. 2656–2660 (2001)

14. Horita, T., Takanami, I., Mori, M.: Learning algorithms which make multilayer neural networks multiple-weight-and-neuron-fault tolerant. IEICE Trans. Inf. & Syst. E91-D(4), 1168–1175 (2008)
15. Sum, J.P., Leung, C.S., Ho, K.I.J.: On-line node fault injection training algorithm for MLP networks: Objective function and convergence analysis. IEEE Trans. Neural Networks and Learning Systems 23(2), 211–222 (2012)
16. Ho, K., Leung, C.S., Sum, J.: Objective functions of online weight noise injection training algorithms for MLPs. IEEE Trans. Neural Networks 22(2), 317–323 (2011)
17. Ho, K.I.J., Leung, C.S., Sum, J.: Convergence and objective functions of some fault/noise-injection-based online learning algorithms for RBF networks. IEEE Trans. Neural Networks 21(6), 938–947 (2010)
18. Sum, J.P.F., Leung, C.S., Ho, K.I.J.: On objective function, regularizer, and prediction error of a learning algorithm for dealing with multiplicative weight noise. IEEE Trans. Neural Networks 20(1), 124–138 (2009)
19. Murray, A.F., Edwards, P.J.: Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training. IEEE Trans. Neural Networks 5(5), 792–802 (1994)
20. Nishimura, K., Horita, T., Ootsu, M., Takanami, I.: Novel value injection learning methods which make multilayer neural networks multiple-weight-and-neuron-fault tolerant. In: Proc. CSREA Int'l Conf. on PDPTA, pp. 546–552 (July 2009)

Framework for Ensuring Runtime Stability of Control Loops in Multi-agent Networked Environments

Nikolay Tcholtchev and Ina Schieferdecker

Fraunhofer FOKUS Institute for Open Communication Systems, Berlin, Germany
{nikolay.tcholtchev,ina.schieferdecker}@fokus.fraunhofer.de

Abstract. The idea of autonomic computing, and accordingly autonomic networking, has drawn the attention of industry and academia during the past years. An autonomic behavior is widely understood as a control loop which is realized by an autonomic entity/agent that manages some resources, in order to improve the performance and regulate diverse operational aspects of the managed network or IT infrastructure. Self-management, realized through autonomic behaviors, is an appealing and dangerous vision at the same time. On one hand, it promises to reduce the need for human involvement in the network and system management processes. On the other hand, it bears a number of potential pitfalls that could be even dangerous to the network, the IT infrastructure, and the corresponding services. One of these pitfalls is constituted by the stability of the control loops, and correspondingly by the interference among multiple autonomic agents operating in parallel. In this paper, a novel approach to ensuring runtime synchronization and stability of multiple parallel autonomic control loops is presented. We formally model the problem of runtime action synchronization, propose different possible solutions, and provide a case study, as well as different performance measurements based on a prototype that implements our approach.

Keywords: Multi-Agent Systems, Autonomic Networks, Stability, Control Loops.

1 Introduction

The continuously increasing complexity of network and systems management has provoked a discussion on the possibility to extend management processes into the device architectures. That way, a certain degree of autonomic decision making within the network or IT infrastructure in question is enabled. This can be best realized by introducing autonomic software agents inside the managed equipment. These agents monitor the surrounding networked environment, exchange information, and undertake corrective or regulative actions. A number of models aiming to realize autonomic control loops have been proposed in the past decade. Initially, IBM introduced an entity called Autonomic Manager [1]. An Autonomic Manager is responsible for managing particular resources via a control loop consisting of the following phases – Monitor, Analyze, Plan, and Execute (MAPE). Furthermore, the

FOCALE [14] [15] architecture introduced a two-layer hierarchy of control loops. The lower of these hierarchies is technology specific - analogue to the Element Management System in traditional Network Management. The higher hierarchy is concerned with the management of the overall system - similar to the traditional Network Management System. Additional initiatives trying to go for an Autonomic Management of future networks, IT infrastructures, and services include CASCADAS [13] with the concept of ACEs (Autonomic Communication Elements) [16], and ANEMA (Autonomic Network Management Architecture) [9][49]. ACEs were used in the CASCADAS project for the management of services (e.g. video streaming services) and implement a control loop, which is inspired by the one described in the initial IBM white paper for autonomic computing [1]. The intelligence of these control loops is given as executable plans. The ANEMA architecture provides a framework that uses utility functions in order to derive behavioral policies which can be executed during the operation of the network. All these approaches address differently the problem of synchronization and ensuring the stability of multiple parallel autonomic control loops. These synchronization and stability aspects constitute a vital issue, since in a complex environment or architecture involving several control-loops executing in parallel, there is an inherent challenge to ensure that the autonomic elements' behaviors are synchronized towards a common goal. This is required in order to avoid a situation whereby each autonomic agent is working towards its own goal, but the overall set of actions/policies degrades drastically the performance and dependability of the system. Such a situation could even result in unwanted oscillations and instabilities of the control loops, leading to a decrease in the provided quality of service.

Coming back to the existing options for achieving action synchronization within the above listed initiatives, a number of approaches can be easily identified. For example, since CASCADAS ACEs operate based on executable plans, it is straight possible for the autonomic system's developer to embed stability constraints in the resulting (collaborative) behavior(s). The utility function based approach for deriving policies of ANEMA suggests that the resulting behaviors will be intrinsically stable and conflict free during the operation of the network. In addition, [27] has investigated possibilities to design collaborative control loops such that they operate in a stable manner. This includes the application of game theory concepts during the design phase, and the use of model driven engineering techniques. The latter can be achieved through different tools or combinations of tools (tool chains), which are applied during the development phase of the overall set of autonomic entities, such that their control loops are intrinsically stable and synchronized by design. The set of applicable tools includes modeling environments such as GME [4], and EMF [35], simulation and verification tools such as UPPALL [36], and finally code generation tools. Examples for tool chains or standalone tools enabling the model driven specification and design of autonomic agents are given in [17] as well as in [18]. Thereby, the approach in [17] is based on the interplay between existing tools facilitated through a model sharing system called ModelBus, whilst [18] [19] [20] and [21] aim at defining specific modeling languages that enable the (stable) design and specification of self-managing entities. In addition, the synchronization between control loops from traditional control theory was widely investigated. Examples of such research, based on game theory and conducted in the scope of cloud computing, is given by [44] [45] and [46]. Moreover, [5] introduces a hybrid approach to the synchronization of multiple self-organizing agents/entities. This hybrid approach is based on the use of so called *archetypes* which are architectural templates embedded inside the software agents in question. The archetypes constraint and control the runtime self-organizing behavior of the entities in a way that the agents work collaboratively and do not contradict. The hybrid nature is given by the fact that different archetypes are specified during the design phase and are dynamically instantiated according to the emerging situation. The authors of [48] have focused on integration patterns for components of autonomic management systems. These patterns semi-formalize the way existing autonomic agents can be brought to work together. Thereby, the integration patterns set the path towards conflict-free interaction of the control loops belonging to the autonomic components being integrated[1]. [31] and [33] propose a methodology to synchronize different independent autonomic control loops by introducing an additional "on top" layer. This additional layer contains a *coordination manager* utilizing finite-state-machines which are specifically developed in a way as to synchronize the actions resulting from the underlying control loops. [42] introduces a synchronization approach for multiple autonomic control loops based on the use of a common knowledge base. That way, independent control loops can share information about their operation and avoid conflicting situations. Finally, the current paper capitalizes on and extends another concept that has been initially presented as part of [27] and [26]. This is the concept of runtime action synchronization for multiple independent parallel running control loops - including policy actions and control theory type of loops. The novelty of this idea and progress beyond state of the art is that it allows, in a generic way, for autonomic agents, which were not intrinsically designed as to collaborate, to synchronize their tentative actions and work together towards improving the performance of the network or IT infrastructure in question. Based on a model regarding the impact of potential actions on selected key performance indicators (KPIs), as well as the importance of these KPIs, we propose and evaluate methods and techniques for the selection of an optimal subset of tentative actions, which are intended for execution by independent autonomic agents.

The proposed framework is complementary to self-organization approaches (e.g. self-organizing maps) where the autonomic agents collaboratively try to achieve an optimal set of reactions without referring to an arbiter for the sake of conflict resolution. We argue that in a real world environment, it is best to combine the framework presented here with traditional self-organization techniques embedded inside the autonomic agents in question.

In order to show the feasibility of the proposed concepts, our approach is validated based on a prototype that allows conducting a representative performance and scalability analysis.

[1] In that context, the approach towards action synchronization presented in this paper can be largely map to the *Hierarch pattern* of [48].

The rest of this paper is organized as follows: Section 2 introduces the architectural aspects of our proposed framework, and presents a case study with respect to how a specific architecture for autonomic networking can be extended by components implementing our approach. Section 3 formally defines the problem of runtime action synchronization and arrives at a mathematical model that allows for applying different types of algorithms for the purpose of ensuring the runtime stability of parallel autonomic control loops. Section 4 reformulates the problem in a way that it is more efficiently solvable, and presents a machine learning approach to obtaining some of the vital parameters for the resulting new model. Section 5 relates our proposed mechanisms to traditional concepts from the area of Control Theory. The following section describes some technical details and design decisions regarding our prototype implementation. Thereafter, based on the described prototype, section 7 describes the experimental setup and results related to the performance of the proposed techniques, in terms of synchronization quality and scalability. The next section presents a case study that demonstrates how the proposed techniques can be applied in the domain of autonomic networking. Finally, section 9 concludes the paper and outlines potential future research directions.

2 Architectural Aspects

This section elaborates on the architectural aspects with respect to the problem of runtime action synchronization.

2.1 Architectural Setup for Runtime Action Synchronization

We propose to introduce an entity that can be requested by other agents to allow or disallow tentative actions based on the goal of optimizing a set of key performance indicators. This results in an architectural setup as the one illustrated in figure 1. The entity denoted as Action Synchronization Engine (ASE) is requested by a number of independent agents - designed and implemented without intrinsic synchronization - to check their tentative actions for potential conflicts, and to inform them back on whether particular actions are allowed or disallowed for execution. Specifically, we consider plain actions (e.g. reset a network interface card) and policy enforcements - an *if(condition)-then(action)* - as (management) actions, since each of those items can be seen as an action (make/enforce) in the corresponding context. Indeed, the ASE is expected to act as an arbiter for negotiation that removes some overlapping and possible contradictions in the actions. Hence, an ASE component would allow only those management actions to proceed that are beneficiary for the overall fitness of the managed infrastructure.

Based on the architectural principle in figure 1, we propose two possible realizations of an ASE component. An ASE can be either put in place as a separate agent, i.e. as a standalone process solely responsible for the synchronization of autonomic control loops executed in parallel, or as a sub-component inside an agent that is primarily designed to manage some operational resources. In the latter case, the

ASE sub-component is started, orchestrated, and configured by the responsible autonomic entity. Clearly, an autonomic agent hosting an ASE sub-component may also refer to it for the sake of synchronization and conflict resolution regarding parallel control loops. This is exemplified in the next section, where we show how one of the emerging architectures for autonomic networking can be extended in a way as to realize action synchronization functionality for its control loops.

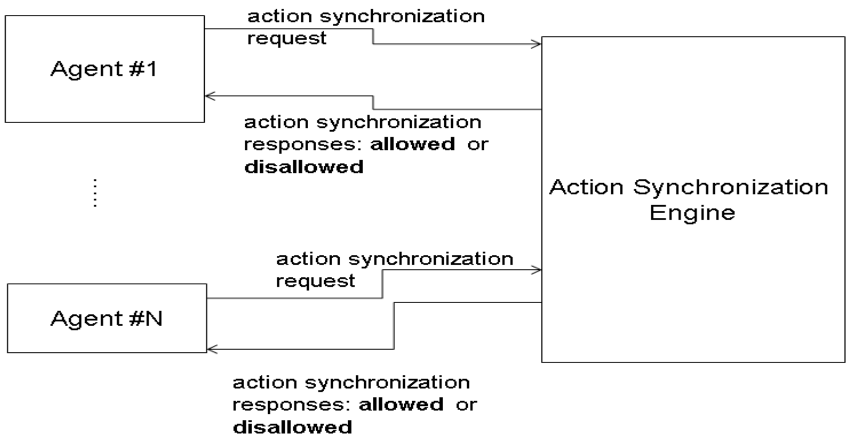

action synchronization request
Agent #1
action synchronization responses: **allowed** or **disallowed**
Action Synchronization Engine
action synchronization request
Agent #N
action synchronization responses: **allowed** or **disallowed**

Fig. 1. Basic architectural Setup for Runtime Action Synchronization

2.2 Case Study: Ensuring Runtime Stability of Autonomic Control Loops in the Generic Autonomic Network Architecture (GANA)

In this section, a case study is presented that illustrates how ASE components can be deployed along a specific architecture for autonomic networking. First, we give an overview of the key features of the architecture selected for our case study, and then propose the necessary extensions.

For illustrative purposes, the Generic Autonomic Network Architecture (GANA) [3] has been selected, since it tries to create a complete view - of the node and the network as a whole - with respect to autonomic self-management of future networks. In addition GANA is currently being worked on at ETSI as to mature towards a reference model for Autonomic Networking [47]. In addition, an initial implementation of the GANA architecture was provided within the EFIPSANS project [6].

GANA defines generic autonomic elements for each required networking function, e.g. routing, forwarding, mobility etc. The core concept in GANA is that of a generic autonomic entity denoted as Decision Element (DE). A Decision Element (DE) executes the logic of a control loop using the management interfaces of its assigned Managed Entities (MEs). That is, a DE is responsible for autonomically regulating the parameters of concretely assigned MEs, and realizes a control loop based on the information it acquires directly from the MEs or from other sources such as embedded network monitoring processes. The information is analyzed, subsequently a decision is made, and an action is executed on the MEs in order to dynamically (re)-configure and regulate their behavior, while striving to achieve a predefined goal. Taking into account that control loops on different levels of functionality are possible, GANA defines the Hierarchical Control Loops (HCLs) framework. In the context of the HCLs framework, four levels exist at which generic Decision Elements and associated control-loops can be designed: (1) protocol-level – autonomic mechanisms within the network protocols, e.g. control loops in OSPF or TCP, (2) functions-level – autonomic control loops responsible for a specific network function, e.g. routing, forwarding, mobility management, (3) node-level - a device as a whole is also considered as a level at which autonomic functions considering the overall node can be implemented, (4) network level – autonomic functions which are executed network wide. Thereby, Decision Elements and corresponding control loops on a higher level manage DEs on a lower level down to the lowest-level MEs i.e. protocols and fundamental mechanisms. Therefore, DEs are designed following the principles of hierarchical, peering, and sibling relationships among each other. These relationships are realized within a node or among the nodes enabling DEs to realize both distributed and centralized control loops. For detailed information regarding the presented concepts, the reader is referred to [3].

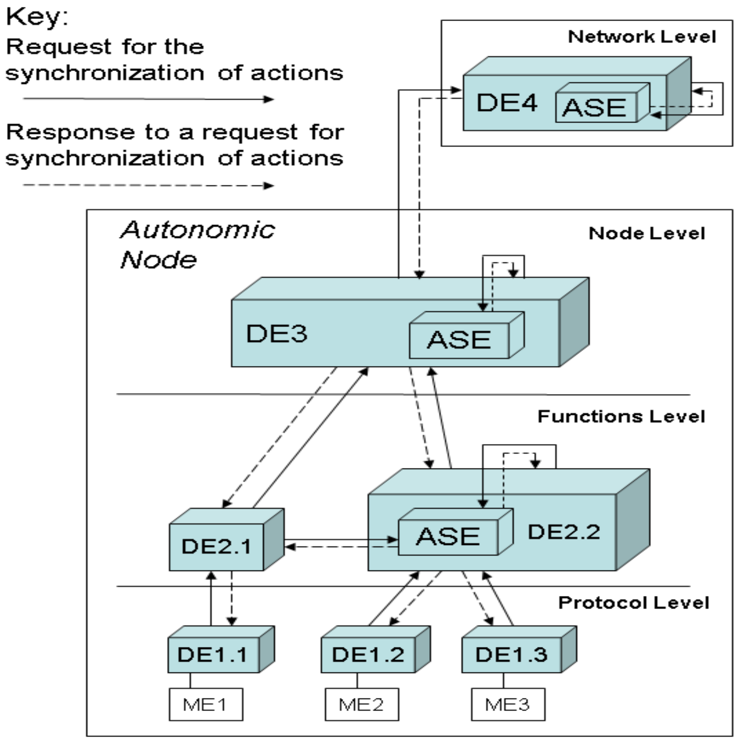


Fig. 2. The extended GANA Architecture

In order to elegantly extend GANA and its generic DEs following the rules defined by HCLs, we propose that ASE components should be embedded inside the corresponding DEs. That is, ASE sub-components should be initialized and started by some of the DEs in order to achieve a runtime configuration as the one depicted in figure 2. An ASE is considered as part of a dedicated generic GANA DE that has been elected or is by design the most appropriate one for acting as an arbiter enabling the negotiation over tentative actions. In that context, the DEs in question can refer to the belonging ASE by using the hierarchical, peering or sibling relations defined within HCLs. Therefore, we require that every DE should keep a list (catalogue) of the actions it is allowed to issue without having to consult an upper level or a sibling DE. As illustrated in figure 2, if a DE (e.g. DE1.1, DE1.2 or DE1.3) faces a problem that is beyond its local scope, i.e. the action to be issued as a response to some challenging conditions is not inside the aforementioned catalogue, it should consult its upper level DE (DE2.1 or DE2.2). The upper level DE should in turn consult the corresponding ASE (hosted by DE2.2) that is expected to resolve potential conflicts and to respond back with a set of actions that are allowed to be executed. After the optimal actions have been selected, the upper level DE informs the lower level DEs in question, whether they are allowed to "fire" some of the actions on their corresponding Managed Entities (ME1, ME2, and ME3). On the other hand, if the upper level DE recognizes that the actions it has been requested to synchronize are beyond its competence, it should further consult its corresponding upper DE (e.g. DE3 or even DE4 in figure 2) on the higher level.

3 The Action Synchronization Problem

This section formally defines the runtime Action Synchronization Problem (ASP). We start with a short introduction and overview of similar efforts in the past years and continue with deriving a model that allows to mathematically reason about the optimal subset of tentative actions requested for synchronization. This model consists of a set of mathematical objects, such as a utility function to optimize, and a set of constraints represented by corresponding matrices and vectors.

3.1 Introduction

In a complex multi-agent environment or architecture involving more than one autonomic control loop that need to execute in parallel, the challenge is to ensure that the autonomic entities' behaviors are synchronized towards a common goal. That is required in order to avoid a situation whereby each autonomic entity is working towards its own goal but the overall set of actions/policies degrades the performance and dependability of the system in question. In order to achieve the aforementioned common goal, we present an approach based on the concept of a utility function that incorporates the state of the network/system, and must be optimized by selecting the optimal subset of tentative actions, resulting from decisions made by the autonomic entities.

Research related to the application of utility functions in the area of Autonomic Computing/Networking has been ongoing since the release of the IBM autonomic computing white paper [1]. The idea of Autonomic Management based on goals, for which utility functions are a way to express, was initially investigated in [12]. [10] is one of the pioneer publications discussing the role of utility functions in self-management. It elaborates on the different classes of policies that can be applied in order to optimize the corresponding utility function and presents a case study carried out in the context of a commercial service management framework. [11] describes a two-level utility function based architecture for monitoring and actively optimizing the performance of an application server. However, the application of the utility functions is mainly embedded inside the autonomic elements and there is no additional arbiter component that handles synchronization requests from autonomic elements, and chooses to allow or disallow specific tentative actions.

3.2 Deriving the ASP Problem

Based on ideas from the theory of Optimal Control [38], we define an optimization problem that should be solved by an ASE agent whenever a set of actions needs to be synchronized. Let Q be a set of performance metrics/KPIs, and $|Q|=n\in N^+$. Let W be a set of weights, each of which is assigned to one of the metrics contained in Q and $|W|= n\in N^+$. These weights represent the importance of a single KPI for the overall health of the system, and thus they all should be positive, i.e. $w_i\geq 0$ for $w_i\in W$, $w_i\in R$ and $i\in\{1, \ldots, n\}$.

Considering a particular point in time t_0, the values of the KPIs in Q are represented by a vector $Q(t_0)=(q_1(t_0), \ldots, q_n(t_0))^T\in R^n$. Suppose that, the higher the values of $q_1(t_0), \ldots, q_n(t_0)$ the better the performance of the system (for KPIs which require to be minimized one can take the reciprocal value), then the following expression gives the system fitness at t_0.

$$SF(t_0) = \sum_{i=0}^{n} w_i q_i(t_0) = \langle w, Q(t_0) \rangle \tag{1}$$

Hence, the goal of the autonomic mechanisms in the node/device and the network is to maximize $SF(t)$ throughout the operation of the system.

Additionally, let A be the set of possible actions that can be potentially issued by the involved autonomic agents, and $|A|=M\in N^+$. By $a_j\in A$ with $j\in\{1, \ldots, M\}$ we denote a single action. Further, the domain relation of an action $d:A\rightarrow\{0,1\}^n$ is introduced. The relation takes as an input an action and returns a (0-1) vector, which contains mappings to the metrics the input action influences if executed. Indeed, if the i^{th} component of the vector is 1, then the i^{th} metric is influenced by the input action, and respectively if the latter is 0, then the metric is correspondingly not influenced. Putting together the domain relation outputs for all actions as columns in matrix form results in what we denote as the domain matrix of A.

$$D = (d(a_1),\ldots, d(a_M)) \in \{1,0\}^{n\times M} \tag{2}$$

The last and most important ingredient for formulating the optimization problem is the impact relation of A – $I{:}QxA\rightarrow R$. The output value of $I(i,j)$ stands for the impact the j^{th} action has on the i^{th} KPI. Thus, if only action a_j is executed at point in time t_0, then the new value of q_i will be $q_i(t_0)+I(i,j)$.

Based on the above definitions, and assuming that in a particular time slice a total number of $m \in N^+$ $(m \leq M)$ actions have been requested for synchronization, the following optimization problem is defined.

$$\max_{p \in \{1,0\}^n} \sum_{i=0}^{n} w_i (q_i(t_0) + \sum_{j=0}^{m} p_j I(i, j)) \text{, } s.t. \; D_m p \leq c \tag{3}$$

In (3), p stands for a (0-1) vector, which gives whether a particular action was allowed to execute or not. Indeed, if the j^{th} position of p is 1, then the corresponding action has been selected for execution; otherwise it has to be dropped. Hence, an optimization with respect to p is equivalent to selecting the optimal set of actions requested in a particular time frame. Moreover, the matrix D_m is a sub-matrix of the domain matrix D of A consisting only of the columns representing the domains of the requested actions. The vector $c \in N^n$ determines the extent, to which actions with overlapping domains are allowed. For example, if $n=4$ and $c=(1,1,1,1)^T$, i.e. only four KPIs are considered, then the additional constraint says that only one action influencing a single metric is allowed. In the case of $c=(2,1,1,1)^T$ two actions are allowed that influence the first metric. Hence, the additional constraint can be used to enforce the resolution of conflicts between actions with overlapping domain regions. Rewriting (3) in vector-matrix form results in

$$\max_{p \in \{1,0\}^n} w^T I_m p + w^T Q(t_0) \text{, } s.t. \; D_m p \leq c \tag{4}$$

where I_m stands for an $n \times m$ real-valued matrix that contains the impact of the requested actions on the considered KPIs. Thus, $I_m(i,j)$ represents the impact of the requested j^{th} action on the i^{th} metric.

Since the term $w^T Q(t_0)$ in (4) is just a constant that reflects the current state of the network, it can be dropped, which is very good news because it means that the values of the KPIs are not needed for the optimization. That fact reduces the overhead produced by the ASE, since no interaction with monitoring functions measuring the KPIs is needed. Hence, the final optimization problem takes the following form:

$$\max_{p \in \{1,0\}^n} w^T I_m p \text{ , } s.t. \; D_m p \leq c \tag{5}$$

The above optimization problem can be interpreted as *"selecting the most appropriate subset of tentative actions such that the change in the state of the system is positively maximized"*.

3.3 On the Complexity of ASP

This subsection investigates around the complexity and the challenges that are expected while solving instances of ASP. First of all, ASP is a special case of integer/binary programming which potentially classifies it as an NP-complete problem, i.e. a hard to tackle problem which can be only solved by a non-deterministic polynomial-time Turing machine. More specifically, ASP corresponds to a special instance of the thoroughly studied 0-1 multi-constraint knapsack problem (MKP) [30]. MKP is defined as follows:

$$\max_{x_i \in \{1,0\}} \sum_{i=0}^{n} g_i x_i \,, \; s.t. \; \sum_{j=0}^{n} v_{ij} x_j \leq C_i \tag{6}$$

with gains (profits) $g_i \geq 0$, volumes (or weights) $v_{ij} \geq 0$ and capacities $C_i \geq 0$ with respect to the overall volume (or weight) for each knapsack. Thereby, $g_i \in R$, $v_{ij} \in R$, and $C_i \in R$. The difference between ASP and MKP is constituted by the ranges for v_{ij} and C_i, and correspondingly d_{ij} - the elements of D_m, as well as c_i - the elements of the constraint vector c. d_{ij} can take only binary values, and c_i can take only positive integer values according to the definition of ASP. Hence, each ASP instance is an instance of MKP. MKP has drawn the attention of the research community for years due to its wide field of application (e.g. resource and budget planning). It is known to be an NP-hard problem [30] with exact algorithms existing for some special cases, as for example described in [32] and [34]. However, to our best knowledge, no exact algorithm exists for high dimensional special instances as constituted by ASP.

These considerations show that finding a solution for the runtime action synchronization problem is not an easy task. Therefore, different heuristics and approximation algorithms can be considered. For example, modern solvers such as GLPK [2] and Coin-OR [8] are quite advanced and would provide a solution that is the best possible based on the underlying optimization algorithm, e.g. branch-and-bound, simulated annealing, tabu search, etc. Moreover, in the next section we propose an approach that allows to partially overcome the obstacles arising due to ASP's computational complexity.

4 Machine Learning Approach to Action Synchronization

In this section, an approach to runtime action synchronization is proposed that allows overcoming the obstacles of inconvenient problem complexity identified in the previous sections. We reformulate the ASP problem and provide a machine learning methodology to handling the uncertain parameters resulting from the reformulation.

4.1 RASP: Relaxation of the ASP Binary Optimization

We start with the previously derived optimization problem, and relax the binary constraints such that the resulting problem belongs to the complexity class P, i.e. to the problems solvable in polynomial time. That is, we turn the condition $p_i \in \{0,1\}$ into an inequality: $0 \leq p_i \leq 1$, for $i \in \{1, \ldots, m\}$. Hence, the new optimization problem is given by:

$$\max_{p} w^T I_m p \,, \; s.t. \; 0 \leq p_i \leq 1 \text{ and } D_m p \leq c \tag{7}$$

This optimization problem is a linear program which constitutes a convex optimization problem. Hence, there is only one optimum and every local optimal solution is also a global one, which means that the diverse optimization techniques will always improve iteratively the quality of the obtained solution. The above formulation constitutes a problem belonging to the complexity class *P*, i.e. efficient polynomial algorithms exist for solving this problem. The result of this optimization is a vector containing values from the interval *[0,1]*. If i^{th} component of this vector is 0 then the corresponding action is disallowed. Correspondingly, if it is 1 then the action should be issued. If an agent, requesting for synchronization, receives as response a value from the interval *(0,1)*, then it can either execute or drop the action, based on an internal threshold θ_i. These thresholds can be supplied by the human experts tweaking the autonomic network. For instance, the history of requests for synchronization can be analyzed offline (e.g. by employing statistical and/or machine learning methods) and appropriate thresholds can be extracted using some optimization tools. In the following sub-section such a methodology is proposed and elaborated in detail.

4.2 Methodology for Obtaining Threshold Parameters for RASP

Assuming that during the operation of a particular system, a history of action synchronization requests was collected $r:=\{r_1, ..., r_{tr}\}$, $tr \in N^+$, a methodology is proposed how to obtain thresholds $\{\theta_1, ..., \theta_M\}$ such that the involved agents are able to decide whether to execute an action or not based on the solutions of RASP obtained by the corresponding ASE component. One can assume that as long as training data is being collected, the system operates based on ASP. The elements of *r* are *(0-1)* vectors, where a 1 is set in case the action at the corresponding position in the vector was requested for synchronization within the particular request, and a 0 in case the action was not requested. The results of the RASP optimization based on the requests in *r* are then given by $X=\{X_1, ..., X_{tr}\}$, $tr \in N^+$ whereby $X_i \in [0,1]^M$. By defining and solving an optimization problem based on the training data in *X*, we would like to obtain the thresholds $\{\theta_1, ..., \theta_M\}$ which are in turn given to the requesting agents and used in the course of runtime action synchronization. This leads to the following maximization objective:

$$\max_{\theta \in [0,1]^n} \sum_{i=1}^{tr} w^T I(X_i - \theta) \tag{8}$$

(8) can be explained as follows: the distances between the threshold values and the RASP results should be optimized in a way that the impact (reflected by these distances) of the corresponding actions is positively maximized. (8) can be further reformulated as follows:

$$\begin{aligned} &\max_{\theta \in [0,1]^n} \sum_{i=1}^{tr} w^T I(X_i - \theta) = \\ &= \max_{\theta \in [0,1]^n} \left[\sum_{i=1}^{tr} w^T I X_i - \sum_{i=1}^{tr} w^T I \theta \right] \\ &\Leftrightarrow \min_{\theta \in [0,1]^n} \sum_{i=1}^{tr} w^T I \theta \Leftrightarrow \min_{\theta \in [0,1]^n} w^T I \theta \end{aligned} \tag{9}$$

The final result in (9) is based on the fact that the first sum in the rearranged maximization is a constant, because it is computed only based on the available training data.

In order to preserve the constraints given in the ASP definition (5), the following constraint is added

$$\sum_{j\in \mathrm{Im}(q_k)} 1_{\{X_{ij}\geq\theta_i\}} \leq c_k \tag{10}$$

with *Im(q*$_k$*)* being the set of actions that impact KPI q_k, and c_k being the bounding value for *k* as given in the "subject to" part of (5). Combining (9) and (10) results in the following non-linear optimization problem:

$$\min_{\theta\in[0,1]^n} w^T I\theta\ ,\quad s.t. \sum_{j\in \mathrm{Im}(q_k)} 1_{\{X_{ij}\geq\theta_i\}} \leq c_k \tag{11}$$

As a further step towards increasing the solvability of (11), one might consider to treat the constraint in a way that it becomes differentiable. That way, optimization algorithms can be applied that explicitly make use of the gradient. A possible direction to go is applying a linear function on both sides of the constraint, since a linear operator would preserve the inequality. A good candidate is the expected value *E[.]* operator, which would also mean that the constraint is relaxed by ensuring that on average the selected subset of actions does not violate the constraint vector. This yields the following:

$$\begin{aligned} & E\left[\sum_{j\in \mathrm{Im}(q_k)} 1_{\{X_{ij}\geq\theta_i\}}\right] \leq E[c_k] \\ \Leftrightarrow\ & \sum_{j\in \mathrm{Im}(q_k)} P(X_{*j}\geq\theta_i) \leq c_k \\ \Leftrightarrow\ & \sum_{j\in \mathrm{Im}(q_k)} (1-P(\theta_i\geq X_{*j})) \leq c_k \end{aligned} \tag{12}$$

Thereby *P(.)* on the last line stands for the cumulative distribution of X_{*j}, i.e. the distribution of RASP results for the *j*th action. Thus, in (12), *P(.)* gives the probability for θ_i to be greater than X_{*j}. Thereby *P(.)*, in terms of a probability model, must be a distribution over the set *[0,1]*. Furthermore, a distribution of X_{*j} can be computed out of the trainings data, e.g. by calculating the maximum likelihood estimators for the targeted model distribution. Combining (9) and (12) results in the following optimization problem:

$$\min_{\theta\in[0,1]^n} w^T I\theta\ ,\ s.t. \sum_{j\in \mathrm{Im}(q_k)} (1-P(\theta_i\geq X_{*j})) \leq c_k \tag{13}$$

The *beta distribution* [41] is a good candidate for *P(.)*, since it generalizes different possible distributions over the *[0,1]* interval. The cumulative distribution function (CDF) of the beta distribution, adapted to the current context for a threshold θ_i , is given by:

$$\mathrm{B}_{X_{ij}}(\theta_i, p_i, q_i) = \frac{\int_0^{\theta_i} t^{p_i}(1-t)^{q_i-1}dt}{\int_0^1 t^{p_i}(1-t)^{q_i-1}dt}, with\ p_i, q_i > 0 \tag{14}$$

(14) has two parameters p_i and q_i, which can be estimated from the training data by using the method of moments [39]. The gradient of (14) can be also easily calculated by taking into account that the derivative of a CDF is given by the corresponding PDF (probability distribution function). This gradient can be used to improve the quality of the obtained thresholds based on algorithms, which can explicitly make use of it for the sake of gradient based optimization.

In the following sections we compare the quality of action synchronization by solving directly the binary optimization problem in (5), and by solving the relaxation based on threshold estimations provided by (11), and by (13) thereby applying (14) as a relevant probability measure. We denote the combination between (13) and (14) as the *beta-distribution approach*. Next, we shortly relate our approach to traditional control theory, and present a prototype implementation, based on which the experiments are conducted.

5 Relating Runtime Action Synchronization to Traditional Control Theory

The ASP problem derived in the previous sections aims at ensuring the stability of an autonomic system implementing multiple control loops. Thereby, the control loops may be based on generic models defined for autonomic computing and networking in the past years (e.g. IBM MAPE [1], FOCALE [14], or GANA [3]) or on control loops as specified within the area of traditional control theory [38]. The latter type of control loops are based on specific transfer functions, and on the state of the system under control in case of feedback loops like the PID (Proportional-Integral-Derivative) controller. Transfer functions can be seen as mathematical artifacts, which describe the required regulative mechanisms based on different parameters such as time or system feedback. Fundamental properties, which the control loop designer tries to achieve by adjusting the parameters of the belonging transfer functions, are the so called SASO properties – Stability, Accuracy, Settling Time and Overshoot [38]. Stability in that case is considered mainly with the oscillations that could be potentially caused by an improperly selected transfer function parameters. Accuracy deals with the degree to which particular desired values of the regulated parameters are achieved. The settling time property can be seen as an indicative for *"how long it takes"* until the effects of the regulative behavior are achieved. Finally, Overshoot gives the maximum deviation (from the desired value) which results from the regulative action until the belonging Accuracy is achieved. Within our framework, we would expect that the underlying control loops are designed considering the SASO properties. The KPI impact of the actions, which result from these control loops, should be modeled as considered after the corresponding settling time. That is, we presume that the final effect of the action is taken into account when modeling the multi-agent environment with the corresponding actions and impact on KPIs. This way the interplay between our approach and traditional control loops is facilitated.

6 Implementation of an Action Synchronization Engine

In order to analyze the technical feasibility of our approach with respect to functionality, scalability and overhead produced by solving the previously derived optimization problems in real time, we implemented an ASE component, which can be started and configured by any "hosting" agent (see section 2), or can operate as an autonomous entity in a multi-agent networked environment. Additionally, an API was defined and developed that can be used by "client" agents willing to synchronize tentative actions resulting from their intrinsic control loops. The implementation was conducted on a Linux platform using C/C++ and POSIX threads. The "client" agents use a specially implemented library for communicating with the corresponding ASE.

The first issue that has to be tackled is related to the solver required for solving the ASP and RASP optimization problems. Research done in the field of electricity spot market optimization problems [7] has compared different open-source solvers for linear and mixed integer programs and gives hints that the SYMPHONY solver (for solving ASP) and the CLP solver (for solving RASP), which are both part of the Coin-OR project [8], perform best. Thus, the ASE implementation was built around the solvers provided by the Coin-OR project. Moreover, the internal architecture of an ASE component consists of two modules – one that takes care of reading and interpreting the configuration data passed to the ASE, and a second one that manages the threads serving the "clients". The configuration data required by an ASE component includes: 1) a file containing the *impact matrix*, 2) a file containing the weights corresponding to the KPIs' significance, 3) a file containing the *constraints* vector that restricts the number of actions that influence overlapping sets of KPIs, 4) the *time interval* after which the solving of the optimization problem is automatically triggered independently of the number of requests, 5) a *maximum number of requests* that automatically triggers the solving of the optimization problem, and 6) a flag indicating whether ASP or RASP should be solved by the ASE entity in order to perform runtime action synchronization. All these parameters need to be supplied to the ASE agent by the time it gets started, and can be provided by a human expert.

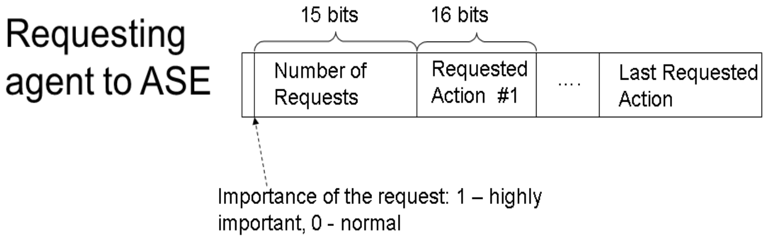


Fig. 3. A Message carrying an Action Synchronization Request

After the module parsing the parameters has completed the configuration of the ASE entity, a main "server" thread is started that listens for, and accepts "client" connections. In parallel a periodical thread is started that "wakes up" up every X seconds (corresponds to the fourth configuration parameter), copies all requests for synchronization, prepares the requests, the impact matrix, the constraints vector etc. in the form required by the native interface of the solver in use, and finally invokes it in order to obtain a solution. Consequently, the relevant parts of the solution are communicated back to the corresponding requesting agents. Apart from the periodical task, every time a connection gets accepted by the ASE component, the requests are stored in a common (for all threads inside the ASE process) registry. In cases that the request was submitted and marked as *highly important* or the *maximum number of requests* after which synchronization is automatically triggered has been reached, the solving of the optimization is invoked, taking into account all synchronization requests stored in the common registry since the last time optimization was triggered. In order to synchronize all the concurrent threads around the common registry and the solving process, POSIX mutexes are utilized.

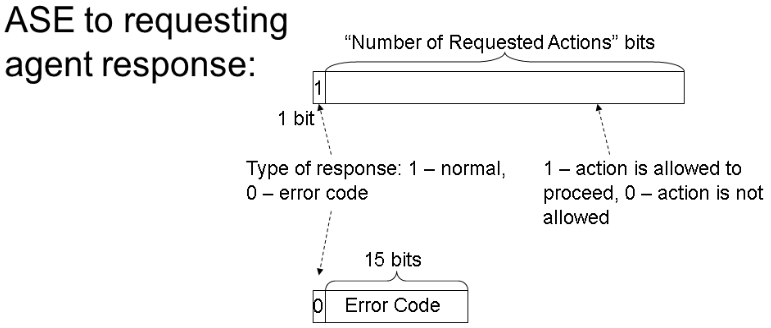

ASE to requesting agent response:
"Number of Requested Actions" bits
1
1 bit
Type of response: 1 – normal, 0 – error code
1 – action is allowed to proceed, 0 – action is not allowed
15 bits
0
Error Code

Fig. 4. Response as a Result of an ASP based action Synchronization Request

The communication between the requesting entities and an ASE component requires the specification of a message format for the exchange of requests and responses in case of an ASP or RASP solving ASE. These messages can be exchanged over Unix Domain Sockets inside a device, or over specially designed protocols for control information exchange between nodes, e.g. ICMPv6. Figure 3 presents the format of an action synchronization request message as implemented in our prototype. The first bit is used to indicate the urgency of the request. The following 15 bits are used to encode the number of requested actions, followed by a set of 16 bit integers representing the requested actions.

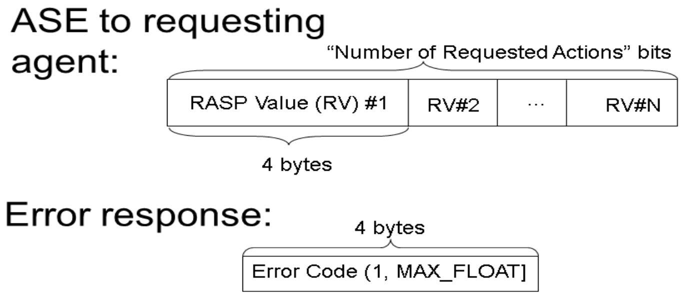


Fig. 5. Response as a Result of a RASP based Action Synchronization

Depending on whether the ASE entity in question is pre-configured to perform an ASP or a RASP optimization, different response formats are required. Figure 4 illustrates the message carrying the response from an ASP based action synchronization. The first bit of the message indicates whether the synchronization was successful or not. In case of a failed synchronization, different error codes can be reported within the next 15 bits. In case of a success, a corresponding number of bits is conveyed which reflect the request for action synchronization thereby indicating whether a tentative action is allowed (bit value 1) or disallowed (bit value 0). Additionally, figure 5 depicts the response message in case of RASP synchronization. It consists of a sequence of thresholds each encoded in 4 bytes. These are meant to be floating point numbers in the range between 0 and 1. In case of failed action synchronization, an error code is returned that indicates the type of problem occurred within the ASE component.

Based on this prototype, the next sections present different numerical evaluations of our approach as well as a case study illustrating its application.

7 Experimental Results

In order to evaluate our approach, we designed a set of experiments such that we can compare the quality of the action synchronization achieved by the different mechanisms and techniques presented so far. Since there are no such systems deployed in practice, we simulated requests and models (including weight vectors, impact matrices, and constraint vectors) of different sizes for our experiments. This allowed us to gain an impression of the performance of the different techniques on a number of models and combinations of tentative actions.

7.1 Qualitative Measurements

The measurements presented in this section are meant to benchmark the quality of the action synchronization procedure, in terms of achieved utility value and constraint satisfaction as determined by the corresponding constraint vector. This utility value is obtained by solving ASP directly, or by solving the corresponding relaxation – RASP, and applying thresholds for accepting for execution or dropping actions.

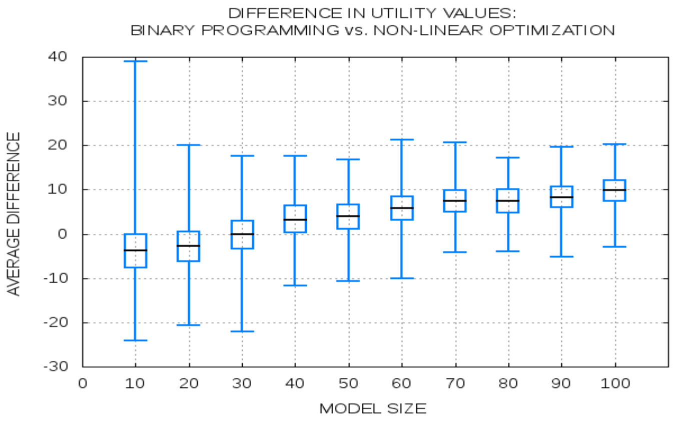


Fig. 6. The Average Difference between the Utility Values obtained by solving RASP combined with applying Thresholds based on optimizing (11), and directly solving the ASP Synchronization

We used models with equal numbers of KPIs and potential tentative actions. For each model size, we took 1000 different model instances. Additionally, 1000 action synchronization requests were simulated as training data for each model size, in order to evaluate the machine learning approach from section 4. Moreover, 100 action synchronization requests were used as test data for each model size, upon which we validated the quality of the different techniques. We developed a Matlab script that implements the machine learning procedures as specified by (11), as well as by (13) with applying the beta-distribution (14) as a probability measure. In the course of this, we made use of the NLOPT [22] library for non-linear optimization. Thereby, we employed first an “Improved Stochastic Ranking Evolution Strategy” [23] to perform a global search for both problems defined in (11) and (13). Afterwards, we conducted a local search based on the “Constrained Optimization by Linear Approximations” [24] algorithm for (11), and on the “Augmented Lagrangian algorithm” [25] for (13), in order to find a precise solution locally. Thereby, the algorithm applied for (13) can explicitly benefit from the derivatives of the utility function and the constraints, which can be computed based on the beta-distribution (14) function. These steps resulted in thresholds obtained based on (11) and (or) (13) for each model instance. The belonging test data was used to evaluate the quality of these thresholds in terms of accepting or dropping an action based on a solution for RASP (7). That is, we first solved RASP, and using the corresponding set of thresholds, we accepted or dropped actions based on the obtained RASP solution vector. The selected actions resulted in a particular value of the utility function, and potentially violated the constraints vector, since the relaxations influence directly the involved constraints. For that reason, on one hand, it is worth to compare the difference between the utility value achieved by using thresholds and the one obtained by solving ASP directly. On the other hand, it is required to measure the magnitude of constraint violation achieved by solving RASP and applying the thresholds. At this point, it is worth mentioning that straight solving the binary program (5) does not lead to any constraint violations, since this is the original version of the optimization problem.

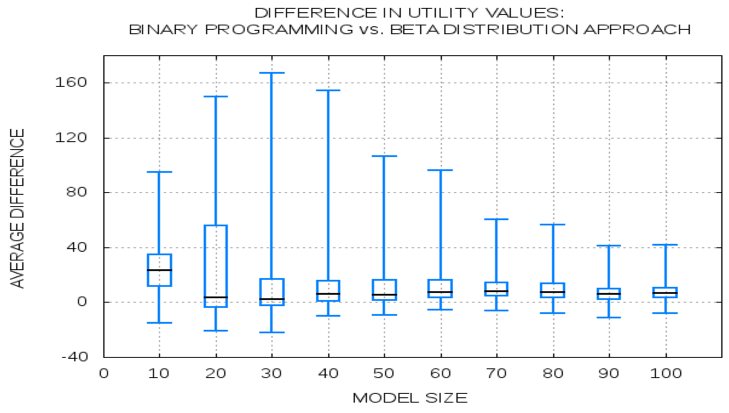


Fig. 7. The Average Difference between the Utility Values obtained by optimizing RASP combined with applying Thresholds based on the Beta-distribution Approach, and solving directly the ASP Problem

Figure 6 and figure 7 illustrate the empirical distributions of the differences between the averaged utility values of the involved approaches for each model size. The averaged utility values were obtained based on the belonging test dataset. That is, we averaged the utility values obtained for each action request in the corresponding test data, and present the empirical distribution of the difference between these averaged utility values for each of the 1000 model instances of the size in question. By difference we mean *"averaged utility value of RASP and (11)" - "averaged utility of ASP"* for figure 6, and *"averaged utility value of RASP and (13)" - "averaged utility of ASP"* for figure 7. Indeed, a positive value within the empirical distribution means that the corresponding RASP based technique performed better than straight binary optimization. The box plots visualize the median, the 25% quantile, the 75% quantile, the minimum and maximum value of the experienced sample. Figure 6 shows that solving RASP, and applying thresholds obtained by (11), performed slightly worse than binary optimization of ASP for smaller model instances. However, with growing model sizes, the RASP based method started performing better even though there were still some outliers (in terms of models) for which binary optimization is better. On the other hand, figure 7 shows that solving RASP, and applying thresholds obtained through the beta-distribution approach, resulted in better utility values for all evaluated model sizes. Again, there were some outliers (in terms of models) by which the binary optimization seems to be the better approach to obtaining optimal system fitness.

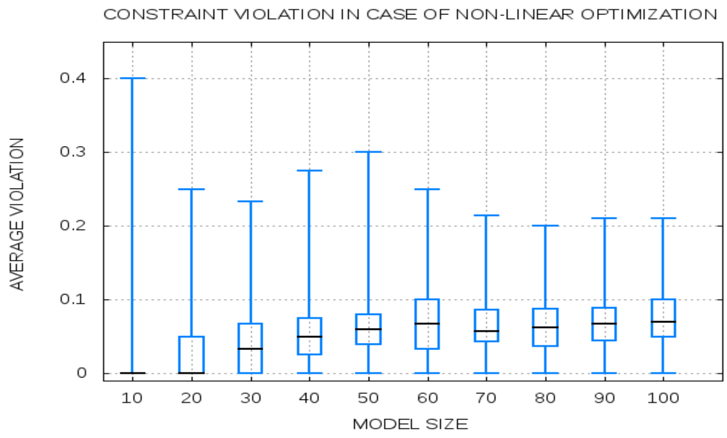


Fig. 8. The Average Violation in case of optimizing RASP and applying Thresholds obtained through (11)

The average numbers of constraint violations while performing RASP based optimization are shown in figure 8 and figure 9. The average value was built based on the test data set for each model instance. Figure 8 and figure 9 clearly indicate that even tough the violations in the case of the beta-distribution approach were minor, the non-linear threshold optimization (11) clearly outperformed the beta approach when it comes to constraint violations. This is not surprising, since (11) reflects to a large extent the initial ASP problem regarding the constraints, while the probability based approach (13) relaxes them by assuring compliance only on average.

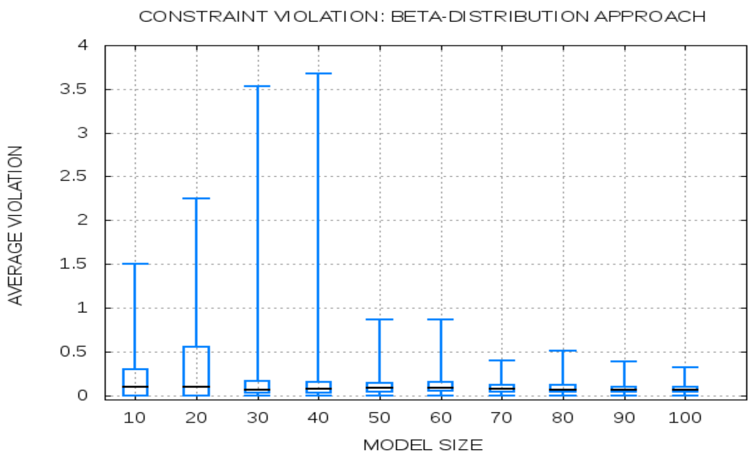


Fig. 9. The Average Violation in case of optimizing RASP and applying Thresholds obtained through the Beta-distribution based Approach

In general, we see that the two discussed RASP based techniques performed reasonably as compared to straight ASP binary optimization when it comes to achieved system fitness, and to satisfying the original constraints, which influence the simultaneous execution of actions.

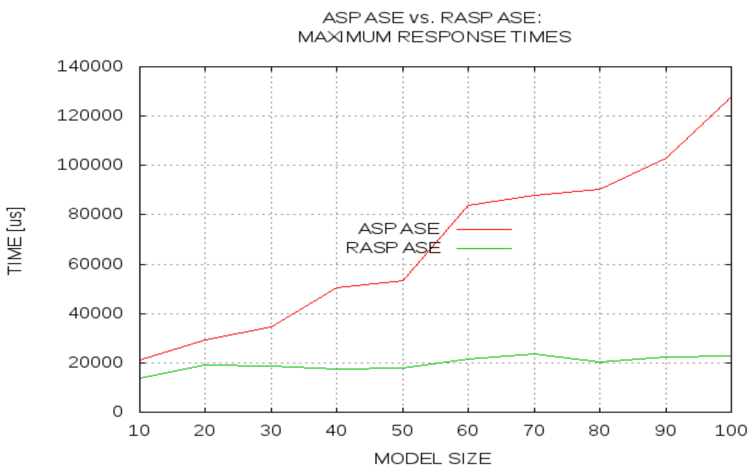


Fig. 10. Maximum Response Time of an ASE Component when solving ASP and when solving RASP

7.2 Scalability and Overhead Measurements

Since ASE entities are meant to facilitate self-management for a large diversity of systems, including network switches and routers, embedded systems, multimedia gateways, and server systems, the issue of scalability and overhead produced by such an agent is of paramount importance. Therefore, we use the previously described prototype to conduct a number of performance and overhead measurements of our proposed approach. In the course of that, we would also like to evaluate the performance characteristics related to the proposed optimization problems, i.e. ASP and the corresponding relaxation RASP.

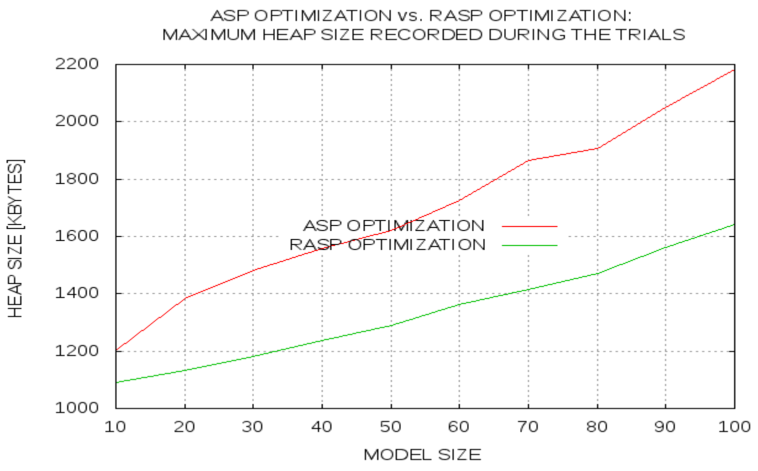


Fig. 11. Memory Consumption of an ASE Component when solving ASP and when solving RASP

The simulations were conducted on a Linux machine with the following hardware parameters: *Intel Pentium 4, 3.00 GHz, RAM 2 GB*. We simulated two requesting agents which were simultaneously issuing a total number of 100 action synchronization requests to an ASE entity. The size of the requests was set to 20 actions requiring synchronization. The last of the requests was sent as *highly important* (see figure 3), such that synchronization is issued immediately for the overall set of actions submitted by both agents. This setup allows also to judge on the scalability of the approach in case a large number of control loops need to be synchronized, since it pushes to the extreme the size of the models and the number of actions to synchronize. The only aspect that is not covered is given by the capability of the underlying hardware and operating system to serve a large number of connections and threads. Potential issues there can be remediated by additional hardware or by increasing the use of self-organization techniques (e.g. self-organizing maps) among the autonomic agents in question.

The maximum of the response times for the two requesting entities and the model size in question is plotted in figure 10. The trend clearly shows that a RASP solving ASE is the better choice in situations where fast responses are vital. Additionally, figure 11 compares the heap memory consumption of an ASP and a RASP solving ASE that can be seen as indicative for the amount of memory consumed while solving the underlying optimization problems. The trend in figure 11 indicates that in case memory is scarce resource, a RASP based approach to action synchronization would be the most suitable one.

8 Case Study: Autonomic Fault-Management and Resilience in Self-managing Networks

Our illustrative case study comes from the domain of autonomic networking and deals with the functions of Autonomic Fault-Management (AFM) and Resilience in self-managing IP networks. Autonomic Fault-Management is understood [28] as an automation of the tasks comprising traditional Fault-Management as indicated by the ITU-T TMN [29] (Telecommunication Management Network) standard. These tasks are: Fault-Detection – *"detect the presence of fault"*, Fault-Isolation - *"find the root cause for the observed faulty condition"*, and finally Fault-Removal - *"remove the identified root cause"*. On the other hand, the resilience of the network depends also on an immediate reaction to an observed faulty condition. This means that an action is required in order to mask the erroneous state until the Autonomic Fault-Management mechanisms have managed to remove the underlying root cause(s). Such an immediate action could be for example the result of a regulative mechanism based on a transfer function as implemented in traditional control theory [38]. Architecturally, these aspects have been addressed in publications such as [43]. The idea is to have two different autonomic entities: one implementing the immediate reaction based on a policy model – resilience agent, and a second one realizing the Autonomic Fault-Management tasks – AFM agent. These entities are introduced in each node of the network. Indeed, when it comes to executing the resulting actions, the two entities need to negotiate in order to resolve potential conflicts due to multiple AFM or fault-masking processes (threads) running in parallel. This negotiation is facilitated by an Action Synchronization Engine within each device. Hence, the case study shows how an ASE can be used to synchronize the tentative actions of autonomic entities within an architecture for Autonomic Fault-Management and Resilience as the one presented in [43].

Figure 12 presents the reference network for our case study. Each of the routers (R1 to R5) is equipped with a resilience agent, an AFM agent, and an ASE agent. Furthermore, each of the routers is running an OSPF routing daemon, e.g. as the one provided by the Quagga routing platform. We focus on potential faulty conditions around R2, and on potential reactions to these faulty conditions issued by the autonomic entities on R2. However, presenting the models and mechanisms that drive the reactions of the autonomic entities is beyond the scope of this paper. For more information, we refer the reader to [37]. In figure 12, R2 is a critical point in the network, since it is a router having different (Gigabit and Fast Ethernet) types of links with different characteristics, as for example MTU (Maximum Transmission Unit) size – 1500 bytes for Fast Ethernet and maximum 9000 bytes for Gigabit Ethernet links. This creates the potential for black hole problems as described in [RFC2923]. These problems are made possible by the suppression of ICMP messages on the router in question having links with different MTUs attached. This is especially critical in IPv6 networks where there is no packet fragmentation on intermediate routers. On the other hand, the (dynamic) suppression of ICMP messages is required in case the network is under attack, since ICMP responses were often used to realize flooding and Deny of Service (DOS) attacks in the past. Hence, it is a good idea to also dynamically regulate the level of ICMP suppression as a reaction to an anticipated attack on the network infrastructure. Thus, the following potential actions, to be issued on R2 by the AFM agent, are introduced: *set-icmp-suppression-(low/medium/high)*. These actions are expected to differ with regard to the subsets of ICMP message types being suppressed on R2. Furthermore, the problem of link flapping can potentially occur on any of the routers including R2. It is constituted by a link that is continuously going up and down thereby affecting the OSPF routing in the network [40], such that the routers have difficulties to converge with respect to available routes. An immediate reaction of the resilience agent to this phenomenon could be given by adjusting the rate of the OSPF Hello timer on the affected routers [40]. Thus, the following potential actions, to be issued on R2 by the resilience agent, are considered: *set-hello-rate-(low/medium/high)*. As a result of the belonging Autonomic Fault-Management processes, the AFM agent can decide to either restart a Network Interface Cards (NIC) on R2, or to restart R2 as whole. This results in the following potential actions *restart-node* and *restart-NIC*. Regarding the KPIs to optimize for the case study, we consider the following QoS metrics from the telecommunications domain: *delay*, *jitter*, *packet loss*, *out of order packets*, and *throughput*, as well as the KPIs of *overall security level*, *CPU utilization* and *memory consumption* on R2.

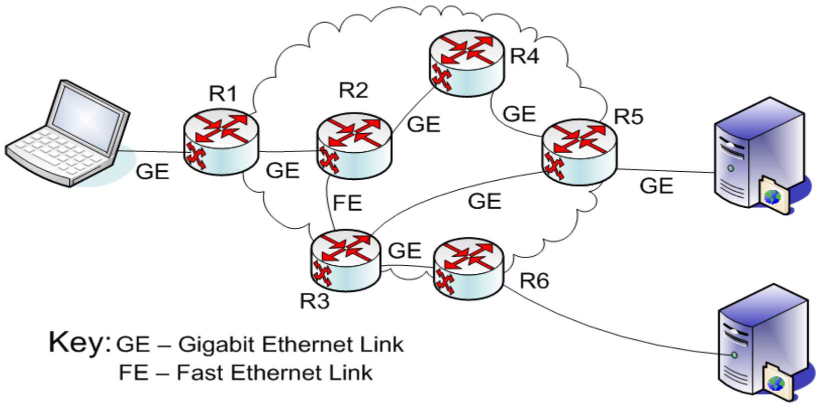


Fig. 12. The Reference Network for our Case Study

Table 1. Model Parameters for the Case Study

	RESTART-NODE	*RESTART-NIC*	*SET-HELLO-RATE-HIGH*	*SET-HELLO-RATE-MEDIUM*	*SET-HELLO-RATE-RATE-LOW*	*SET-ICMP-SUPPRESSION- HIGH*	*SET-ICMP-SUPPRESSION- MEDIUM*	*SET-ICMP-SUPPRESSION-LOW*	*CONSTRAINTS*	*WEIGHTS*
DELAY	5	4	5	1	-2	-3	1	3	2	15
JITTER	4	3	5	1	-2	-3	1	3	2	10
OUT OF ORDER	4	3	5	1	-2	-3	1	3	2	15
PACKET LOSS	5	3	5	1	-2	-4	2	4	2	20
THROUGHPUT	5	4	5	1	-2	-4	2	4	2	20
SECURITY LEVEL	-1	0	-1	1	2	5	1	-5	2	20
CPU	5	0	-4	1	3	0	0	0	2	10
MEM	5	0	-4	1	3	0	0	0	2	10
DISALLOW SOME OF THE ACTIONS TO EXECUTE SIMULTANEOUSLY										
	0	0	0	0	0	1	1	1	1	0
	1	1	0	0	0	0	0	0	1	0
	0	0	1	1	1	0	0	0	1	0
PRIORITIZE SPECIFIC ACTIONS										
	2	1	0	0	0	0	0	0	1	1
	2	0	1	0	0	0	0	0	1	1
	2	0	0	1	0	0	0	0	1	1
	2	0	0	0	1	0	0	0	1	1
	2	0	0	0	0	1	0	0	1	1
	2	0	0	0	0	0	1	0	1	1
	2	0	0	0	0	0	0	1	1	1

Table 1 presents the parameters of the model created based on the case study described hitherto. The first eight columns show the *impact matrix* of the model. The last two columns show correspondingly the constraints and the weights of the model determining the importance of the KPIs to optimize. For the impact of a certain action on the KPIs we take integer values from the interval [-5,5] meaning that a negative degrades the KPI, and correspondingly a positive value improves the KPI. For the weights, values from the interval [0,20] were considered. A number of interesting aspects are given by the lines, not assigned to a KPI. They demonstrate how model parameters can be extended beyond the modeling notions described in section 3, such that *particular* actions do never get executed simultaneously, and *certain* actions are prioritized. In the current case study, it is logically required that only one action at a time is executed from the sets *set-hello-rate-(low/medium/high)*, *restart-(node/nic),* and *set-icmp-suppression-(low/medium/high)* in case multiple actions from these sets are requested for synchronization. This is modeled by the second part of table 1, and is achieved through creating a relationship between these actions, and restricting this

Table 2. Illustrative Action Synchronization Requests

REQUESTED	*RESTART-NODE*	*RESTART-NIC*	*SET-ICMP-SUPPRESSION-LOW*	
SELECTED	*YES*	*NO*	*NO*	
REQUESTED	*RESTART-NIC*	*SET-ICMP-SUPPRESSION-LOW*	*SET-ICMP-SUPPRESSION-MEDIUM*	
SELECTED	*YES*	*YES*	*NO*	
REQUESTED	*RESTART-NODE*	*SET-ICMP-SUPPRESSION-HIGH*	*SET-ICMP-SUPPRESSION-LOW*	*SET-ICMP-SUPPRESSION-MEDIUM*
SELECTED	*YES*	*NO*	*NO*	*NO*
REQUESTED	*RESTART-NODE*	*SET-HELLO- RATE-LOW*	*SET-HELLO- RATE-MEDIUM*	*SET-ICMP-SUPPRESSION-HIGH*
SELECTED	*YES*	*NO*	*NO*	*NO*
REQUESTED	*RESTART-NIC*	*SET-HELLO-RATE-MEDIUM*	*SET-HELLO-RATE-HIGH*	*SET-ICMP-SUPPRESSION-MEDIUM*
SELECTED	*YES*	*NO*	*YES*	*NO*

relationship by extending the impact matrix and the constraints vector. Through this, only one action of each set is selected for execution after a synchronization process. The last segment of table 1 illustrates how an action can be set to have a higher priority than another one. In that case we prioritize the action *restart-node* over each of the other potential actions. This is achieved by an additional row in the *impact matrix* for each intended prioritization. Indeed, *restart-node* is favored based on the new *"impact values"*, and only one of the two actions in question, i.e. *restart-node* and the action with respect to which it is being prioritized, can be selected according to the new *constraints* (refer to the belonging entries in the constraints vector).

To illustrate the operation of an ASE component based on the case study and belonging model, we used our prototype, in order to issue different requests for action synchronization, and observe the resulting selected subsets of actions. Thereby, the ASE agent was operating on the binary program constituting the ASP version of the above described model. Table 2 summarizes some of the results from our experiments. It can be observed that indeed only one action from the sets *set-hello-rate-(low/medium/high)*, *restart-(node/nic),*and *set-icmp-suppression-(low/medium/high)* has been selected for execution, in case multiple actions from theses sets were requested for synchronization. Moreover, we can also see how the action *restart-node,* when requested, is always prioritized over the others. For the rest, the selection is performed based on the impact of the tentative actions on the fitness of the network as defined in (1) and (5). The results in table 2 show how such an ASE component can resolve emergent conflicts between parallel autonomic control loops, thereby always trying to achieve an optimal improvement in the operation of the network.

9 Conclusions and Future Research Directions

This article introduced a framework for ensuring the conflict-free and synchronized operation of multiple parallel autonomic control loops. Such a framework is useful in cases when the agents implementing the parallel control loops are not designed with the explicit awareness of each other, but instead operate in a way as to achieve their own goal and optimize the performance of the resources they were assigned to manage. By introducing components called Action Synchronization Engines (ASE), we propose to have an arbiter that enables the negotiation over tentative actions which are about to be executed in the scope of the parallel running control loops. An ASE component needs a model that allows it to reason about the impact of different actions such that a synchronization procedure can be performed. Therefore, we developed a mathematical model called ASP (Action Synchronization Problem), instances of which can be given to an ASE agent in order to facilitate its operation in a particular context. This model is based on the optimization of some key performance indicators (KPIs). It allows for: 1) specifying the importance of single KPIs, 2) specifying the impact of the potential actions on the KPIs in question, and 3) influencing the execution of actions with overlapping impact domains (in terms of KPIs). Moreover, in the course of the presented case study, we saw how the model can be used to: 4) explicitly disallow the simultaneous execution of some actions, and 5) prioritize some actions over others. Unfortunately, this model results in a problem which is of very hard computational complexity. For that reason, an additional effort was presented that aims at relaxing the original problem in a way that it can be efficiently solved. The resulting formulation delegates some degree of decision to the requesting agents. These decisions are based on parameters (thresholds) for which we proposed a machine learning approach for their configuration. A comparison between the diverse techniques – direct binary optimization and machine learning based decisions – was also conducted. This comparison shows that the proposed reformulation does not degrade the quality of the action synchronization procedure, while at the same time reducing the memory consumption and improving the response time of an ASE component.

The approach proposed in this article opens a number of exciting research and development challenges. For example, the investigation of further ways for modeling runtime action synchronization and belonging efficient algorithms might be an interesting research topic. Besides, there is a need for sophisticated tooling that allows to easily create ASP models as proposed within this article. This could potentially be done in a way that the system's operator does not get involved into the mathematical details presented in this paper, but rather relies on easy to understand atomic artifacts, which can be efficiently combined to enable the self-managing system coping with various emergent situations. Finally, one might also consider the mapping of multiple parameters of the ASP model to business goals, and use this to achieve revenue by for example adapting the behavior of an autonomic network to specific type of expected traffic.

Acknowledgement. This work has been partially supported by EC FP7 EFIPSANS project (INFSO-ICT-215549).

References

[1] Autonomic Computing: An architectural blueprint for autonomic computing, IBM White Paper (2006)

[2] GLPK, GNU Linear Programming Kit, `http://www.gnu.org/software/glpk/` (as of date March 18, 2012)

[3] Chaparadza, R.: Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments. IEC Annual Review of Communications 61 (December 2008)

[4] Bai, J., University, V., Abdelwahed, S.: A Model Integrated Framework for Designing Self-managing Computing Systems. In: The Proceedings of FeBid 2008, Annapolis, Maryland, USA, June 6 (2008)

[5] Debbabi, B., Diaconescu, A., Lalanda, P.: Controlling Self-Organising Software Applications with Archetypes. In: 2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems, pp. 69–78 (September 2012)

[6] EC funded- FP7-EFIPSANS Project, `http://efipsans.org/` (as of date March 18, 2013)

[7] Thorncraft, S.R.: Evaluation of Open-Source LP Optimization Codes in Solving Electricity Spot Market Optimization Problems. In: 19th Mini-Euro Conference on Operation Research Models and Methods in the Energy Sector, Coimbra, Portugal, September 6-8 (2006)
[8] Coin-OR, http://www.coin-or.org/ (as of date March 18, 2013)
[9] Derbel, H., Agoulmine, N.: ANEMA: Autonomic network management architecture to support self-configuration and self-optimization in IP networks. Published in the Elsevier Journal of Computer Networks (November 2008)
[10] Kephart, J.O., Das, R.: Achieving Self-Management via Utility Functions. IEEE Internet Computing 11(1), 40–48 (2007)
[11] Tesauro, G., Kephart, J.O.: Utility Functions in Autonomic Systems. In: Proceedings of the First International Conference on Autonomic Computing, May 17-18, pp. 70–77 (2004)
[12] Lehtihet, E., Derbel, H., Agoulmine, N., Ghamri-Doudane, Y.M., van der Meer, S.: Initial approach toward self-configuration and self-optimization in IP networks. In: Dalmau Royo, J., Hasegawa, G. (eds.) MMNS 2005. LNCS, vol. 3754, pp. 371–382. Springer, Heidelberg (2005)
[13] CASCADAS project, http://acetoolkit.sourceforge.net/cascadas/ (as of date March 18, 2012)
[14] Strassner, J., Agoulmine, N., Lehtihet, E.: FOCALE: A Novel Autonomic Networking Architecture. In: Latin American Autonomic Computing Symposium (LAACS), Campo Grande, MS, Brazil (2006)
[15] Famaey, J., Latre, S., Strassner, J., De Turck, F.: A hierarchical approach to autonomic network management. In: 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, pp. 225–232 (2010)
[16] Höfig, E., et al.: On Concepts for Autonomic Communication Elements. In: Proc. of 1st IEEE International Workshop on Modelling Autonomic Communication Environments (2006)
[17] Prakash, A., et al.: "Formal Methods for Modeling, Refining and Verifying Autonomic Components of Computer Networks. Transactions on Computational Science 15, 1–48 (2012)
[18] Rodriguez-Fernández, C., et al.: Self-management capability requirements with SelfMML & INGENIAS to attain self-organising behaviours. In: Proceeding of the Second International Workshop on Self-organizing Architectures, SOAR 2010 (2010)
[19] Vassev, E.: ASSL: Autonomic System Specification Language - A Framework for Specification and Code Generation of Autonomic Systems. LAP Lambert Academic Publishing, Germany (November 2009)
[20] Vassev, E., Hinchey, M.: The ASSL approach to specifying self-managing embedded systems. Concurrency and Computation: Practice and Experience 24(16), 1860–1878 (2012)
[21] Vassev, E., Mokhov, S.A.: Developing Autonomic Properties for Distributed Pattern-Recognition Systems with ASSL - A Distributed MARF Case Study. Transactions on Computational Science 15, 130–157 (2012)
[22] NLOPT library, http://ab-initio.mit.edu/wiki/index.php/NLopt (as of date March 18, 2012)
[23] Runarsson, T.P., Yao, X.: Search biases in constrained evolutionary optimization. IEEE Trans. on Systems, Man, and Cybernetics Part C: Applications and Reviews 35(2), 233–243 (2005)

[24] Powell, M.J.D.: Direct search algorithms for optimization calculations. Acta Numerica 7, 287–336 (1998)

[25] Birgin, E.G., Martínez, J.M.: Improving ultimate convergence of an augmented Lagrangian method. Optimization Methods and Software 23(2), 177–195 (2008)

[26] Tcholtchev, N., Chaparadza, R., Prakash, A.: Addressing Stability of Control-Loops in the Context of the GANA Architecture: Synchronization of Actions and Policies. In: Spyropoulos, T., Hummel, K.A. (eds.) IWSOS 2009. LNCS, vol. 5918, pp. 262–268. Springer, Heidelberg (2009)

[27] Kastrinogiannis, T., Tcholtchev, N., Prakash, A., Chaparadza, R., Kaldanis, V., Coskun, H., Papavassiliou, S.: Addressing Stability in Future Autonomic Networking. In: Pentikousis, K., Agüero, R., García-Arranz, M., Papavassiliou, S. (eds.) MONAMI 2010. LNICST, vol. 68, pp. 50–61. Springer, Heidelberg (2011)

[28] Li, N., Chen, G., Zhao, M.: Autonomic Fault Management for Wireless Mesh Networks. Electronic Journal for E-Commence Tools and Applicatoins (eJETA) (January 2009)

[29] The FCAPS management framework: ITU-T Rec. M. 3400

[30] Fréville, A.: The multidimensional 0-1 knapsack problem: An overview. European Journal of Operational Research 155, 1–21 (2004)

[31] Mak-Karé Gueye, S., de Palma, N., Rutten, E.: Coordinating energy-aware administration loops using discrete control. In: Proc. of the 8th International Conference on Autonomic and Autonomous Systems, ICAS 2012 (March 2012)

[32] Gorski, J., Paquete, L., Pedrosa, F.: Greedy algorithms for a class of knapsack problems with binary weights. Computers & Operations Research 39, 498–511

[33] Gueye, S.M.-K., Rutten, E., Tchana, A.: Discrete Control for the Coordination of Administration Loops. In: 2012 IEEE Fifth International Conference on Utility and Cloud Computing (UCC), November 5-8 (2012)

[34] Gilmore, P.C., Gomory, R.E.: The theory and computation of knapsack functions. Operations Research 14, 1045–1075 (1966)

[35] EMF, `http://www.eclipse.org/modeling/emf/` (as of date March 18, 2012)

[36] Hessel, A., Pettersson, P.: Model-Based Testing of a WAP Gateway: An Industrial Case-Study. In: Brim, L., Haverkort, B.R., Leucker, M., van de Pol, J. (eds.) FMICS 2006 and PDMC 2006. LNCS, vol. 4346, pp. 116–131. Springer, Heidelberg (2007)

[37] Tcholtchev, N., et al.: Scalable Markov Chain Based Algorithm for Fault-Isolation in Autonomic Networks. In: 2010 IEEE Global Telecommunications Conference, GLOBECOM 2010, pp. 1–6 (2010)

[38] Hellerstein, J.L., et al.: Feedback Control of Computing Systems. Wiley-IEEE Press (September 2004) ISBN: 978-0-471-26637-2

[39] Schlittgen, R.: Einführung in die Statistik. 9. Auflage. Oldenbourg Wissenschaftsverlag, Oldenbourg (2000) ISBN 3-486-27446-5

[40] Wang, F., et al.: A Route Flap Suppression Mechanism Based on Dynamic Timers in OSPF Network. In: Proceedings of ICYCS 2008, pp. 2154–2159 (2008)

[41] Farnum, N.R.: Some results concerning the estimation of beta distribution parameters. J. Oper. Res. 38(3), 287–290 (198)

[42] de Oliveira Jr., F.A., Sharrock, R., Ledoux, T.: Synchronization of multiple autonomic control loops: Application to cloud computing. In: Sirjani, M. (ed.) COORDINATION 2012. LNCS, vol. 7274, pp. 29–43. Springer, Heidelberg (2012)

[43] Tcholtchev, N., Grajzer, M., Vidalenc, B.: Towards a Unified Architecture for Resilience, Survivability and Autonomic Fault-Management for Self-Managing Networks. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 335–344. Springer, Heidelberg (2010)

[44] Charalambous, T., Kalyvianaki, E.: A min-max framework for CPU resource provisioning in virtualized servers using H-infinity Filters. In: 2010 49th IEEE Conference on Decision and Control (CDC), December 15-17, pp. 3778–3783 (2010)
[45] Kalyvianaki, E., Charalambous, T., Hand, S.: Resource Provisioning for Multi-Tier Virtualized Server Applications. Computer Measurement Group Journal (CMG Journal 2010) (126), 6–17 (2010)
[46] Kalyvianaki, E., Charalambous, T., Hand, S.: Self-Adaptive and Self-Configured CPU Resource Provisioning for Virtualized Servers Using Kalman Filters. In: 6th Int. Conference on Autonomic Computing andCommunications, ICAC 2009 (2009)
[47] ETSI AFI, `http://portal.etsi.org/portal/server.pt/community/afi/344` (as of date March 18, 2012)
[48] Frey, S., Diaconescu, A., Demeure, I.: Architectural Integration Patterns for Autonomic Management Systems. In: 9th IEEE International Conference and Workshops on the Engineering of Autonomic and Autonomous Systems (EASe 2012), Novi Sad, Serbia, April 11-13 (2012)
[49] Agoulmine, N.: Autonomic Network Management Principles. Academic Press (December 2010) ISBN: 0123821908

Part II
Geometric Modeling and Simulation

3D Reconstruction from Planar Contours: Analysis of Heuristic Tiling Approaches

Marcelo da Silva Hounsell, Leonardo Kandler Bittencourt, and Alexandre Gonçalves Silva

LARVA –LAboratory for Research on Visual Applications,
PPGCA – Graduate Program in Applied Computing, DCC- Department of Computer Science,
UDESC – Santa Catarina State University, Joinville, SC, Brazil
{marcelo,alexandre}@joinville.udesc.br, leonardokb@gmail.com

Abstract. 3D reconstruction from planar contours using the heuristic approach implements rules for three different phases of the process: correspondence, tiling and branching. We have analyzed existing algorithms and have been able to isolate their constituent pieces which allowed us to foresee many other possible (atomic) contributions. These pieces of algorithms were implemented independently and different mixes were compared using performance, geometrical and user-centered metrics. It was found that user-centered analysis are not reliable; that local criterion do not reflect on the whole model´s quality and; that there is a tradeoff between performance and geometrical metrics. We could also find a particular mix of algorithm pieces that lead to a novel 3D reconstruction algorithm. Moreover, we have built an open source freeware framework where more mixes can be composed and where further testing and improvements could be carried out.

Keywords: Solid Modeling, 3D Reconstruction, Tiling Algorithm, Planar Contour, Heuristic Approach.

1 Introduction

Advances in technology allowed a variety of non invasive exams to produce visual information about an object-of-study (human body, ground, solid objects, etc.). These exams can be applied to areas such as medicine, petroleum industry, geology, manufacturing metrology, quality assurance, animation/gaming industry and, so on. All of them benefit from a given set of planar parallel images that represents cross-sections taken from the object (hereafter called slices). These slices are processed to show contours of interest that highlight the object constructs (such as flesh and bones on a medical application). However, the biggest advantage is the underlying 3D information that can be processed to compose a proper 3D model for visualization and analysis purposes. This process, called "3D Reconstruction", becomes more important as the use and popularity of non invasive exams increase.

3D Reconstruction algorithms can be achieved through a three phased approach known as the Heuristic Approach [10] which relies on meaningful rules to perform its task. These phases are: “Correspondence”, which aims to identify which contour in successive planar slices, should be connected to each other; “Branching”, which focus on defining how multiple contours can be connected altogether, and; “Tiling”, that cares to compose the actual tiles that builds the surface of interconnected contours in-between slices. By identifying separate phases, one can study optimizations for each one of them. This “divide to conquer” strategy facilitates improvements for the whole 3D Reconstruction algorithm.

We took the “divide to conquer” strategy further down to the tiling phase. In doing so, five steps could be identified and many alternative solutions could be elected for each step and their mix were explored in search for an improvement of the resulting reconstructed model. To evaluate the resulting models, measurements were taken regarding performance, geometrical features (of visual origin but geometrical nature) and user´s perception of a good model. Therefore, this paper aims to analyze the tiling phase (how it work and how it could be improved), the reconstructed geometrical model (how to objectively evaluate it) and, users perception of the reconstructed model (how user´s perceive the results).

The paper is divided as follows: Section 2 discusses how to evaluate reconstructed models; Section 3 reviews existing tiling approaches; Section 4 presents tiling phase as a collection of steps, places existing solutions on a general framework and presents new contributions for each step; Section 5 proposes an improved tiling approach and compare its features with existing approaches regarding performance, geometrical and user-centered metrics; Section 6 concludes this text.

2 Evaluating Reconstructed Models

The quality of the 3D reconstructed model can be related to its visual features compared to the expected appearance of the corresponding real object. However, the real model could not be available to perform the comparison once it could be a person´s internal organ (rather, the objective of the whole process is to estimate the shape and look of the unavailable object). But, this is an intuitive and subjective [3] analysis that cannot be computed automatically. Rather, the quality of the reconstructed model can be taken from its geometrical features.

Existing approaches [10, 12] optimize the tiling phase in belief that the efficiency of the process according to a given local criteria yield a good "model". Local criteria are used to assess individual geometrical elements (faces, edges, points). Local and global optimization criteria based on volume, area, edge length (line segments of the contour on the slice as well as segments in-between slices) and, others have been surveyed but found pour performance and unsuitability even for trivial variations (a weird two opposite pyramids shape happens too often)[10]. Therefore, it seems useful to find a positive correlation between (a) subjective user´s resemblance perception and (b) objective geometrical measures from the resulting model.

It should be clarified that the literature differentiates the edges of a reconstructed model that lie on the planar slice from an edge that goes in-between slices: the former are called "segments" and the latter, "spans". Span size and amount are a direct consequence of the tiling approach applied. Therefore, we argue that the quality of the reconstructed model could be related somehow to the spans of the model.

Geometrical measures include: number of vertices in the model; edge-related metrics: minimum and maximum edge size, average and standard deviation of edge size, minimum and maximum span size (which takes into account only those edges that connect vertices from the contours of different planar slices - [6, *apud* 11]), average and standard deviation of span size, and; minimum area, maximum volume.

- Minimum area have been considered the best, more usual and, the metric that usually produce a good result [11]; achieving the minimum area for the whole resulting model [6] is easy to compute and can be done during ongoing tiling processing;
- Maximum volume can also be calculated during tiling processing [7].
- Shortest average size of the edges can be identified during tiling processing and aims to construct small triangles in the model [4, 5, 8].

Volume-based criteria have no sound motivation and give the impression that it could somehow inflate or deflate the resulting model. Area-based criteria do not seem appropriate because a same value for area can be reached for a completely different underlying mesh of faces. Edges are one of the most basic geometrical features of a model; it bares close relationship to the mesh of faces and can be easily computed. Models that comply with the common sense notion of “neat” would present a mesh of faces that shows a regular distribution and little or no visual twist effect. These features can be visualized when the 3D reconstructed model is rendered in flat shading.

Shortest average size of edges alone do not relate directly to model's quality because it can be misleading once maximum and minimum values could be very different for the same average value. Models that have spans bigger than necessary tend to appear as being twisted to some of the corresponding slices. Therefore, once span size is a measure that is related either to the tiling process as well as to the final model's visual features, a thorough analysis on this metric can lead to a unifying way to assess 3D reconstruction.

Minimum or maximum span sizes alone do not guarantee a regular neither twisting-free mesh. Minimum average span sizes do not guarantee regularity because a given average value can be drawn for different min/max values. Median span size, although better represents the expected size of the spans on the model, do not represent how far apart the minimum and maximum values of the span are. Minimum median span size accounts for the best predictable size of spans which reflects faces with smaller edges and, thus minimized twisting. Standard deviation of span sizes represents how it deviates from that given expected value. Therefore it better reflects the regularity of the faces on the mesh.

All above mentioned metrics are geometrical-related and will be used in order to objectively assess the model´s features. However, performance measures should be taken because more heuristics tend to consume more processing time. Furthermore, as we are dealing with visual aspects, users can be asked to assess the model´s quality directly but this is of subjective nature and should be taken with caution.

3 Existing Heuristic Tiling Approaches

The problem statement for tiling is as such: given a set of points that describe two contours in successive planar slices, how to build a mesh of faces using these points as vertices in order to reconstruct the original model? The following sub-sessions address tiling approaches for performing mesh generation:

3.1 Local Proportional Dispersal

Li, Ma and Tan [9] proposed that the number of points of each contour (say contours N and M) should be equal, so that there is a one-to-one match while constructing the edges of the mesh. For that end, points are added in the contour that has a smaller number of points to the amount of the difference in points between the contours.

The tiling occurs as follows: The two closest points between the adjacent contours are defined as the initial references. Then, linear versions of both contours are created (l_1 and l_2) which must have the same direction. The linearization keeps the proportional distance between points by calculating the Euclidean distance between them and projecting it in a linear segment.

Afterward, the linear version of all M points at one contour is parameterized to obtain a sequence of T values ranging from 0 to 1 and the same is done to obtain a sequence of the N points at the other contour to generate S, where $T = (t_1, \cdots, t_M)$ and, $S = (s_1, \cdots, s_N)$, M > N. Then, for each segment in S the closest sequence in T is sought, as shown in Figure 1.

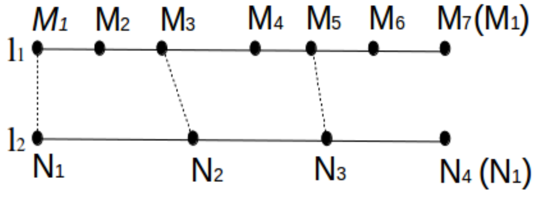


Fig. 1. Linearized contours and initial points linked [9]

Note that the correspondence between the points in l_1 and l_2 is given by the parameter-based proximity and not by accuracy. For instance, M_3 could have t = 0.33 and N_2 would be considered close enough to s = 0.36, allowing a small error to accumulate in the calculation of the proportionality.

Points that are inside the identified segment are proportionally transferred to S based on the local distances of M_1-M_3 and N_1-N_2, hence this approach was named Proportional Local Dispersal. In the case, one mismatched point (M_2) in the l_1 segment was found and a proportional counterpart needed to be created in l_2 (with its corresponding new s value). At the end, (M - N) points are added to l_2 so that the two contours can be matched one-to-one. The actual mesh is generated by replicating a pattern of triangles connecting M_i and N_j points. To implement this algorithm, [9] assumed that the contours have some degree of concentricity and the points that define the contours are oriented in the same direction; otherwise it will generate an object that looks like two opposite pyramids. Note that this approach first calculates 3D distances between pairs of points but the remaining comparisons are done in the 1D domain which saves computational time.

The main contribution of this approach is the points balancing step which creates a proportional counterpart of M into the N contour. This distribution guarantees that regions of a contour with point's concentration will be matched to similar concentration of points in adjacent contour. But for contours with huge difference in the number of points, it adds an equivalent computational cost.

3.2 Shortest Span

Chen et al. [4] created a triangular-shaped connection between the closest points of contours regardless the amount of points in each contour. As in the previous approach, it starts off by identifying the points that have the shortest 3D Euclidean distance. The subsequent triangles are formed by choosing the shortest span built either incrementing the index of the target point in the contour M or N. The procedure continues until all points at both contours were connected to form triangles.

This approach does not generate any extra point, but calculating a 3D distance between points generates a considerable computational cost. Also, if the amount of points of the two contours is different, a point on the contour of a smaller number will probably connect to various points at the other contour.

3.3 Balanced Distance

Anzolin, Hounsell and Silva [1] also performed point balancing between contours but they added the number of points of one contour into the other so that, at the end, both contours end up with (M+N) points. They considered that despite increasing the number of points two fold, the more points are distributed on the contours, the better the final mesh will be. It surely takes more time as well so; they used a much simpler heuristic to distribute the points: for every two consecutive points of a contour, the planar 2D distance between them is calculated and every new point is added in the middle of the biggest segment (the pair that has the longest distance). This distribution strategy minimizes point concentration in a specific part of the contour so balancing the number of points and balancing the distance between them as well, which contributes to produce a more regular mesh.

To define the starting points for the actual mesh generation, a bounding box of each contour is calculated and the points of each contour that is closest to a chosen corner will be elected the starting points. The triangles are then generated similarly to [1]. This approach doubles the points of the resulting model which also add a considerable computational cost but tends to generate regular meshes.

4 Methodology: Dissecting Tiling

To achieve the objectives of the tiling phase, authors have attempted *ad hoc* adjustments (called normalization [10] or, contour matching [15]) that accounts for position, size and orientation changes in the input data (the contours). We consider that these adjustments (and others) are part of the process. This was made clear when we took a "divide to conquer" strategy, pretty much like dissecting the problem, where we isolated constituent contributions (algorithms). This analysis led to an in depth view of the heuristic approach of the tiling phase.

As a result, we propose that the tiling phase is subdivided into five steps: contour offsetting, contour orientation, contour twisting, points balancing and, points linking. Each one of these steps has received some attention and indeed there are some existing solutions for each one but when analyzed in isolation from each other, it can be seen that other alternative solutions are also possible.

Fig. 2 summarizes all five steps of the tiling approach and their alternative solutions. Some existing solutions can be seen (see references that appear in Fig. 2) but we proposed many others to be taken into account for future tiling approaches. Therefore, a tiling approach can now be thought of as a mix of 5 constituent solutions, which will be discussed following:

- **Contour Offsetting.** This step adjusts the contours on their own slices (the XY plane) so that they become aligned (say, concentric) to each other when seeing from the third dimension (Z). Size adjustments can also be performed at this stage. If applied, the working contours must be changed back to original position and size for the remaining tiling steps. For the contours to become concentric, some sort of contours' center calculation must be done to drive the translation. This step is optional but when present, it minimizes contour offset influence while calculating distance between points of adjacent contours;
- **Contour Orientation.** This step evaluates if all points that represent the contour are laid at the same clock or counterclockwise orientation according to a chosen convention. This is required if there is no guarantee that all contours are organized in a standardize way;
- **Contour Twisting.** This step deals with reordering the points of the contours so that their initial referential points are the first point for mesh generation and for further processing. This reorganization can be done in two steps: first, finding which point will be the reference point, and; second, reordering the points in the internal data structure;
- **Points Balancing.** This step is optional but, if included, tends to result in an object with faces of more uniform size. It is concerned with the way that the number of points of the contours is balanced, i.e. it will change position or add new points. If adding points, it governs how they are distributed on the contour;
- **Points Linking.** This step effectively determines which pair of points from the contours should be linked in order to become an edge of the polygonal mesh.

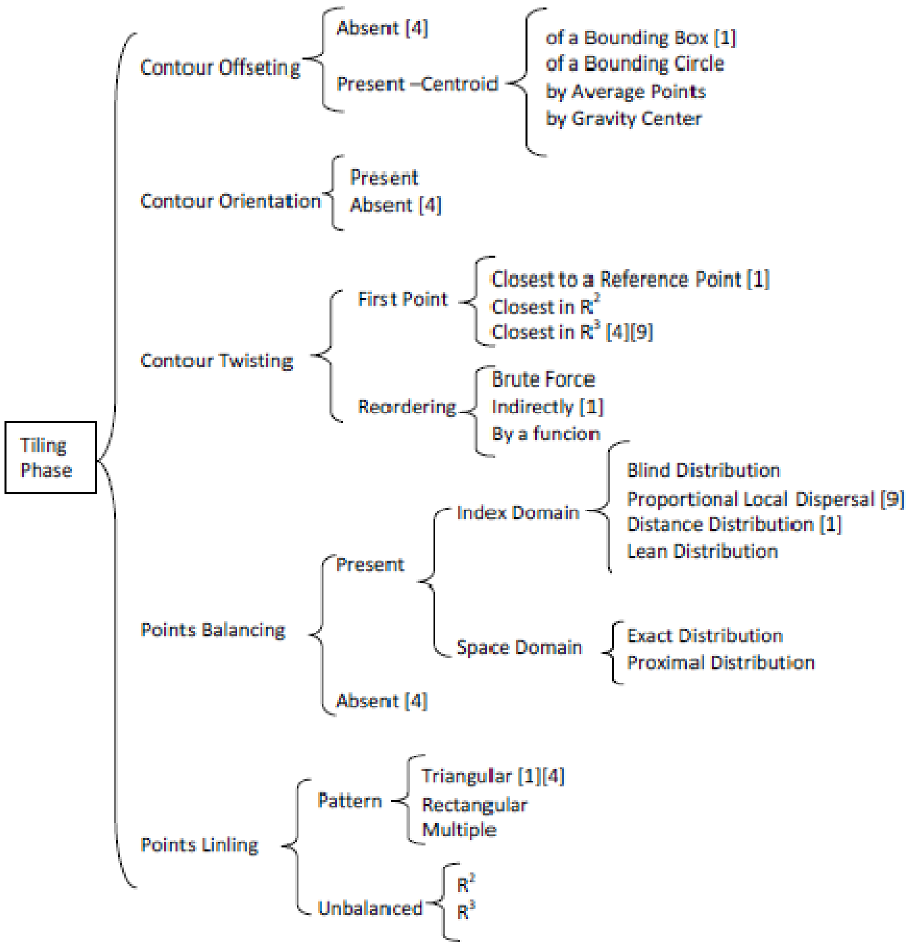


Fig. 2. An Overview of Tiling Steps

The following sections will present novel solutions to every step mentioned above.

4.1 Solutions for Contour Offsetting

The repositioning of contours is dependent on the calculation of the shape's center. To calculate the center of a contour, the center of gravity, arithmetic average of points, its bounding box center or, its bounding circle center, amongst others, can be used:

- To calculate the **center of gravity** of a polygon, one must divide it into triangles and take an average of their center of gravity weighted by their area. The division of the polygon into triangles is required once a triangle´s center of gravity is trivial to calculate. Because of these three calculations (polygon division, triangle center and weighted average) it is of great computational cost. However, the center of gravity is more meaningful and is of interest to many other physical applications;

- Calculating the **average** of all vertices of the contour is of low computational cost, but may be affected by contour´s features and points distribution. For example, the concentration of points in a corner of the contour causes an undesirable shift of the center towards that end;
- The center of an envelope is a fast approach which is not influenced by the distribution of points in the contour. The center of the contours´ **bounding box** [1] is an alternative as well as its **bounding circle** (which coincides with its center). The choice between them would depend on which one promotes the best fit to the actual contours´ shape.

4.2 Solutions for Contour Twisting

The most common way to find the initial points of two contours is calculating the closest points in 3D [4][9] but this would require comparing each point to all others. Another less expensive way is electing an auxiliary and external reference point and then finding the closest point to it but in the same slice, i.e. in the 2D plane [1]. After identifying the reference point, it is useful to reorder the points in the data structure so that they become the first. To reorder the points the following were identified:

- **Brute force:** Reorders the data structure (a vector of points that represent the contour) by effectively moving the points. This option requires preprocessing all contours at the beginning but gives a performance gain during run-time;
- **Indirect way:** The initial point divides the data structure into two halves; it is simply a matter of producing another vector inverting the order of the two halves [1]. This technique also requires preprocessing but gives a performance gain during run-time;
- **Function:** Instead of changing the actual position of points in the data structure, a reordering function that recalculates the required position on the original vector can be used. In this technique, there is no preprocessing, and thus has no computational costs at the beginning but requires extra computation at run-time for every access of a point in the vector.

4.3 Solutions for Points Balancing

This step is performed when it is required that the two contours should have the same number of points because either twisting or linking steps (explained later) would produce a clearer relationship between the points. This relationship allows issuing a predictable tessellation pattern of faces (triangles or rectangles).

The points balancing step is divided into two types: in the index domain or, in the space domain. The former uses indexes of the vector that stores the points of the contour to reason about the insertion of new points or modifying an existing one. The later takes into account the actual point, either in the R^2 (2D) or R^3 (3D) space, in order to calculate the positioning of new points.

- **Proportional Exact Distribution:** A variation of [9] was devised so that rather than just creating points in-between segments of bigger edges of one contour into the other, it replicates all mismatched parametric values of one curve into the other (unless they are considered too close for a given allowance). Thus the error that would accumulate in the selection of the bigger edge of a contour no longer occurs and exact points´ proportionality is guaranteed. For this end, after obtaining the two linearized index sequences *T* and *S*, the union of their values creates just one sequence *R* which will hold for both of them. The correspondence one-to-one will be obtained as new points in the M as well as in the N contours are generated and all of them will have a counterpart. The final object will have a doubled number of points which demands a longer processing time but the resulting object has smoother surface (due to consequent higher number of faces). This approach improves the proportional distribution of points in both contours, without the need to calculate distances in R^3 Euclidean space.
- **Proximal Distribution:** Another way to propagate the pattern of distribution of a contour into another is to, instead of generating new points (the amount of DIF (M, N) as in [9]) by the parametric equation of segments in search for proportionality, project a mismatched point in the M contour onto a line segment in the N contour in R^3. This projection ensures that the resulting point is the closest equivalent (or proximal) missing point in the other contour and therefore will be linked to this one later on. This alternative was devised to deal with the fact that even when ensuring proportionality in the index domain, the original algorithm can generate points far apart in the Euclidean space. The shortest distance is sought within the respective segment but if the projected point is not in the desired segment, the original Proportional Distribution [9] can be used. In Fig. 3, M has seven points and N has 4 points at the beginning (see dots). Three new points need to be created. The squares represent two new points created through the projection of points of M on the proper line segment in N.

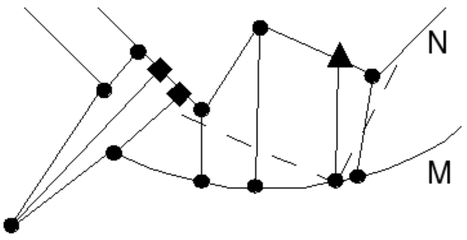


Fig. 3. Projecting Points of one Contour into Another

The triangle represents a point created by the proportional distribution on the index domain because there is a mismatched point that does not have a projection in the corresponding segment of N, as the dotted lines suggest. The number of points generated by this technique is equivalent to that proposed by [1], but is a different way of **balancing** points.

- **Lean Distribution:** Analyzing the correspondence of points proposed by [1] where extra points are added on the longest segment, it was noted that the balancing can be faster by reducing the number of points generated in each contour. Applying [1] solution results in a total of 2(M + N) points. If only the difference of points between M and N is added to N there will be only 2M points in the final model (supposing M is bigger than N).
- **Blind Distribution:** Considering M > N, the blind distribution inserts n points in N, where n is **given** by Eq. 1. This strategy aims to distribute an equal set of points among segments without considering any other factor (distance, for example, hence the name "blind").

$$n = (\text{Div}(M,N) - 1) + \text{Mod}(M,N) \qquad (1)$$

Blind distribution is performed in two steps: the first step spreads an equal amount of extra points in the middle of all segments of N (thus the first term in Eq. 1 accounts for the integer difference between M and N) and; the second step ensures the same amount of points between the two contours by spreading the fractional remains of the division over the existing segments (as in the second term of Eq. 1). This solution is fast because avoids calculating distances in 2D or 3D Euclidean spaces. Figure 4 shows two contours where M has eight points and N has three points. Therefore one point (Div(8,3) -1 = 1) is to be created in all three segments of N (seethe dots). The remainder is 2 and these points are represented by squares, spread over the first two segments.

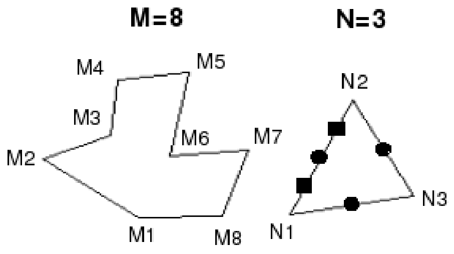


Fig. 4. Blind Distribution of Points

4.4 Solutions for Points Linking

There are two solutions for points linking: the first is to create spans (edges between contours) with the smallest possible size in order to create triangular shaped faces. The required size calculation can be performed using 2D or 3D distance. This strategy works whether the outline is balanced or not. Calculating distances in R3 is more computationally expensive than doing it in R2, although of the same computational complexity. A second solution can be applied if the points are balanced, where patterns of faces can be defined.

Figure 5 (a) shows that either triangular faces (such as the $M_2N_1M_1$ face) where one point can end up connecting to multiple points or, quadrangular faces (such as the $M_5N_3N_2M_4$ face) can be generated; Figure 5 (b) shows a triangular pattern when the set of points are balanced. Figure 5 (c) shows a rectangular pattern of faces created for balanced set of points (like $M_1N_1N\backslash M_2$).

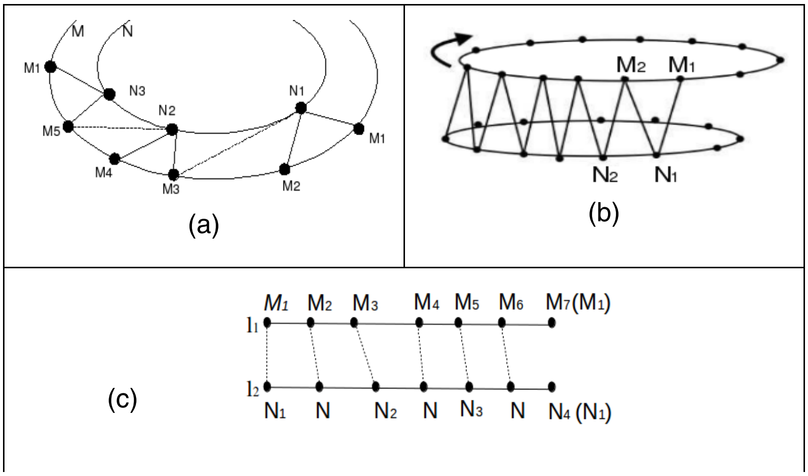


Fig. 5. Patterns of faces according to balanced contours

4.5 Mixing Solutions to Obtain a (new) Tiling Approach

A tiling approach can be devised by mixing solutions from every one of the five steps. There are 3240 possible combinations (mixing every atomic solution to any other from Fig. 2 there will be 5x2x3x2x9x6 of them) and some of the solutions are dependent from others (for instance, some Points Linking solutions would require any Points Balancing solution beforehand).

After gathering some experience on the performance of each solution, we found that contour offsetting, points balancing and points linking were the steps of major consequences for performance and quality of the 3D reconstructed model. Then, new tiling approaches were defined by mixes of these three steps. Four promising approaches were selected from preliminary tests and are further detailed. These were named Delta1+, Delta2+, Delta3+ and Delta4+.

Table 1 presents the mix of solutions for all tiling approaches found in the literature as well as for all Delta alternatives. For simplicity existing approaches will be referred to by acronyms: AHS for [1], CCCH for [4] and LMT for [9]. It can be seen that existing approaches can be described as a specific mix of solutions for the five steps presented above as all their solutions were the founding idea for that step and have been implemented therein.

Table 1. Composing Tiling Approaches from Step Solutions

Parameter/Algorithm		AHS	CCCH	LMT	Delta1+	Delta2+	Delta3+	Delta4+
Curve Twisting	First Point	Closed to Aux. Point	3D-Distance	3D-Distance	3D-Distance	3D-Distance	2D-Distance	3D-Distance
	Reordering	Index	Index	Index	Index	Index	Index	Index
Curve Orientation		No	No	No	No	No	No	No
Curve Offsetting		No	No	No	Average Extreme x and y	No	No	Average Extreme x and y
Point Linking		Triangular	3D-Distance	Quadrangular	Quadrangular	3D-Distance	Quadrangular	3D-Distance
Point Balancing		Distance Distr.	No Balancing	Local Proportional Distr.	Exatly Distr.	Exatly Distr.	Exatly Distr.	Exatly Distr.

5 Tests and Results

Four data sets (veins, femur, heart and lungs) were used to be reconstructed using the three reference algorithms as well as four Delta mixes. These four models were selected because they were used by other works and their raw data were made freely available at [2]. Figures 6 and 7 show the resulting 3D models of veins and femur reconstructed using 4 approaches. All models were rendered using flat shading to highlight the face mesh. Table 1 shows the metric data for reconstructing the veins and Table 2 shows the data for reconstructing a femur. Fig. 8 shows two approaches to reconstruct a heart and a lung.

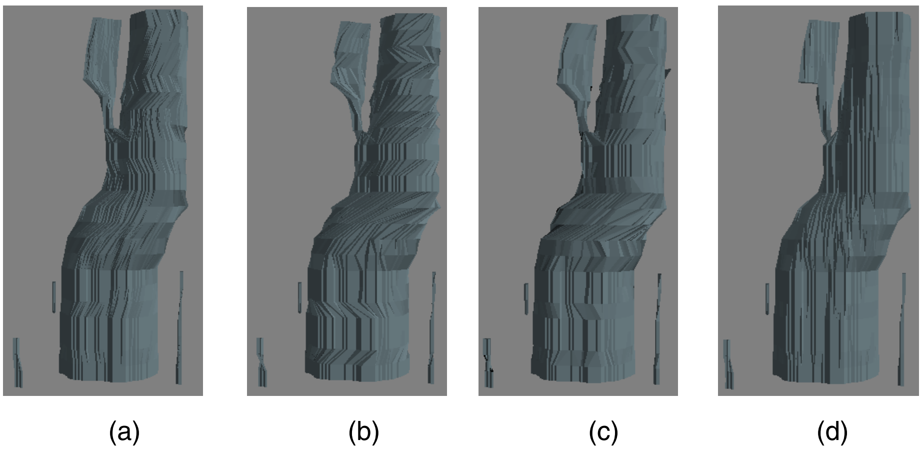


Fig. 6. Veins reconstructed using AHS (a), CCCH (b), LMT (c) and Delta4+ (d) approaches

All four models have 1285 points distributed over 65 contours but on a different number of slices. For instance, data used for veins (see Fig. 6) were distributed over 19 slices and for the femur (see Fig. 7), over 30 slices.

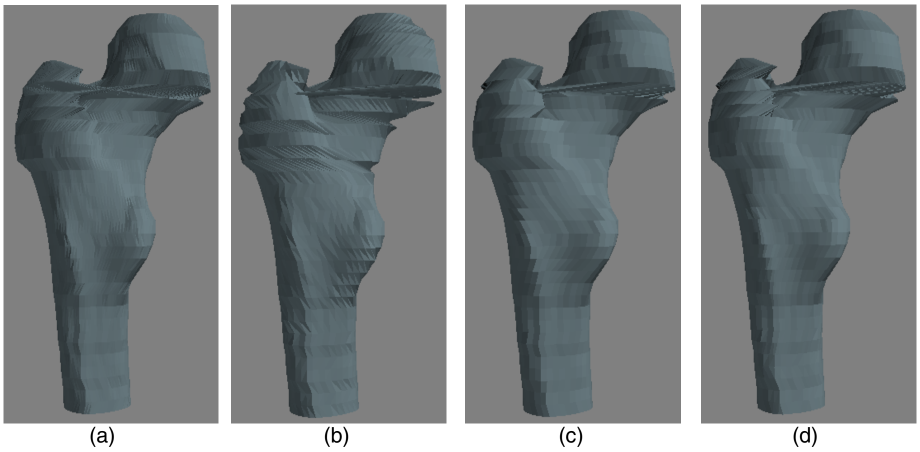


Fig. 7. A femur reconstructed using AHS (a), CCCH (b), LMT (c) and Delta4+ (d) approaches

The metrics gathered from applying the algorithms for the veins models in Fig. 6 are presented in Table 2 and include: total time spent; the amount of points and spans for the resulting model, and; the statistics (average, median, maximum, minimum and standard deviation) for generated spans. Table 2 also indicates the best values (where a "↑" symbol can be seen), the second best values (where a "<>" symbol can be seen) and, the worst values (where a "↓" symbol can be seen). Note that best values not always means smaller values and the similar can be said about the worst values.

Table 2. Measurements taken from spans and segments of the veins model

Vein								
Approach	Time (ms)	Final Points	Spans	Span				
		Amount	Amount	Average	Median	Minimum	Maximum	Standard Deviation
AHS	870	6691↓	32436↓	22,641663	21.256960◊	20.010153↓	93.4764635◊	5.996489◊
LMT	598◊	1545◊	11332↑	23,013941	22,828427	10,163136	101,998324	9.642954↓
CCCH	455↑	1285↑	16218◊	24.294483↓	22.66929↓	11,000000	106.927088↓	8,653944
Delta1+	1379	5731	19704	21.103474◊	21,669078	10,000043	101,686636	8,074481
Delta2+	1415	5731	29556	23,175017	21,333267	20,000000	101,652875	6,919721
Delta3+	1249	5731	19704	21,298615	21,693989	10.000043◊	101,686636	8,268876
Delta4+	1508↓	5731	29556	20.272964↑	20.755141↑	10.000030↑	44.867962↑	3.366304↑

It can be observed that despite generating almost the same number of points, AHS did so in a shorter time, but Delta4+ achieved the smaller mean and smaller standard deviation span sizes, i.e., their spans are smaller and vary less according to the median. Delta4+ showed a flaw in the upper left corner of the channel due to the other phase of the reconstruction algorithm, the correspondence (not discussed here).

All data were gathered from an AMD Turion X2 2.0 MHz computer with 4GB of RAM, 3D Nvidia Gforce 7150M graphics card and Linux Ubuntu 9.04 32bits operating system. Java programming language and a Java 1.5 compiler were used for the implementation. Additional data were gathered considering sizes of all edges (not only spans). The same pattern distribution for minimum and maximum values were obtained while using spans only. Therefore, it was confirmed that there is no need to include the size of segments because they always sum up exactly the same (the perimeter of the contour) regardless the tiling strategy. Thereafter, only span size metrics were considered here forth.

Table 3 shows the measurements taken while reconstructing a femur, shown in Figure 7. For this model CCCH approach was the fastest and produced fewer number of points but, Delta4+ resulted on smaller values for average, median and standard deviation of span sizes.

Table 3. Measurement taken from spans of the model of a femur

Femur								
Approach	Time (ms)	Final Points	Spans	Span				
		Amount	Amount	Average	Median	Minimum	Maximum	Standard Deviation
AHS	496	4666↓	13524↓	0,160681	0,109915	0.100000↓	1,335871	0,155199
LMT	239◊	1492◊	3588↑	0,178202	0,109736	0,100000	1,547527	0.205473↓
CCCH	238↑	1285↑	6226	0.189202↓	0.124599↓	0,100000	1,197815	0,158996
Delta1+	975↓	4594	6690	0,167041	0,108792	0,100000	1.619757↓	0,185243
Delta2+	831	4480	12431	0.148482◊	0,110335	0,100000	1.017957◊	0.120808↑
Delta3+	802	4480	5799◊	0,148967	0.107879◊	0,100000	1.017723↑	0.123733◊
Delta4+	864	4594	13380	0.134890↑	0.104511↑	0.099999↑	1,409906	0,132289

Fig. 8 shows the reconstruction of a lung from a set of 19 slices and of a heart from 30 slices for the AHS and Delta4+ approaches.

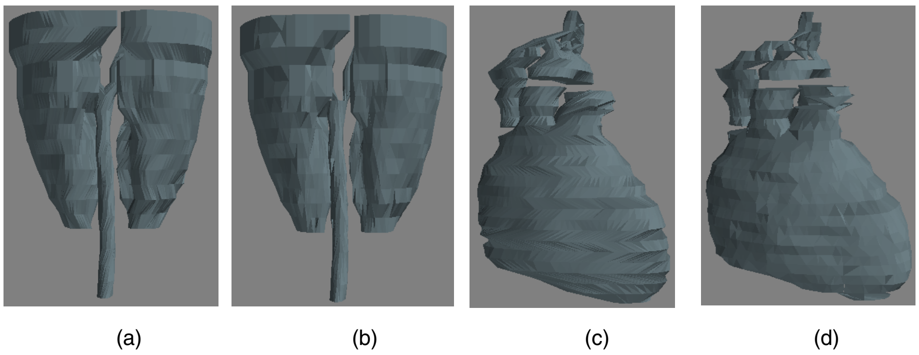


Fig. 8. 3D Reconstructed models of a lung and a heart

Measurement for models in Fig. 8 can be seen in Tables 4 and 5, respectively. Images (a) and (c) are the result of the AHS algorithm and, (b) and (d) are the result for the Delta4+ mix. The approaches showed the same pattern of results as before, i.e.: CCCH is the faster and produces the minimum amount of extra points; Delta4+ produces the minimum span, the minimum average span, the minimum median and the minimum standard deviation of span sizes; LMT produces the minimum amount of spans, and; AHS produced the shortest maximum span.

Table 4. Measurements taken from spans of the model of a lung

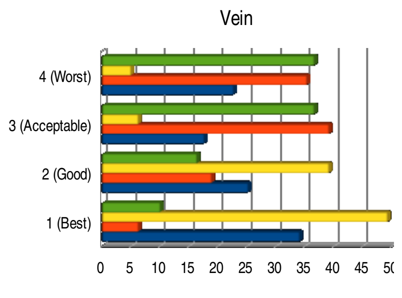
Lung								
Algorithm	Time (ms)	Final Points	Spans	Span				
		Amount	Amount	Average	Median	Minimum	Maximum	Standard Deviation
AHS	242	3936↓	10604↓	120.459357◊	110,780752	100,000000	318.632233↑	27.705468◊
LMT	162◊	1474◊	2840↑	129,822049	111,266060	100,000000	432,152751	45,302336
CCCH	87↑	1285↑	5302	182.402978↓	142.305877↓	100,000000	810.977188↓	108.034075↓
Delta1+	473	3704	5070	132,415928	110.704216◊	100,000000	468,027526	53,149214
Delta2+	432	3704	9391	134,968528	114,312337	100,000000	468,027526	52,809167
Delta3+	426	3704	4476◊	134,269309	112,042078	100,000000	468,027526	54,507152
Delta4+	473↓	3704	10140	110.852217↑	105.135464↑	100,000000	394.405882◊	18.895067↑

Table 5. Measurements taken from spans of the model of a heart

Heart								
Algorithm	Time (ms)	Final Points	Spans	Span				
		Amount	Amount	Average	Median	Minimum	Maximum	Standard Deviation
AHS	241	3962↓	10708↓	0,192038	0,144682	0,100000	0.698352↑	0.108137◊
LMT	78◊	1534◊	2926↑	0,214455	0,150147	0,100000	1.304033↓	0.171052↓
CCCH	77↑	1285↑	5354	0.223873↓	0.167227↓	0,100000	1,135772	0,145252
Delta1+	333	3820	5112	0,191630	0,136035	0,100000	0,913895	0,132578
Delta2+	315	3782	9575	0,191436	0,138006	0,100000	0,913895	0,131086
Delta3+	349	3782	4577◊	0.189780◊	0.136721◊	0.100012↓	0,913895	0,132512
Delta4+	368↓	3820	10424	0.150981↑	0.115733↑	0.099999↑	0.701373◊	0.089206↑

6 User Assessment

The 3D models of the veins, femur and lungs were reconstructed using all tiling approaches previously presented as well as Delta4+. Reconstructed models were placed side-by-side on a web page and displayed in a way to be seen all together. They were shuffled, labeled "A" to "C" and presented to 79 subjects which were asked to: "Please rank all 4 models of the given object indicating the letter that correspond to the "1-best model", "2-good model", "3-acceptable model" and "4-worst model". Subjects were mostly IT undergrad students at various stages of their courses as well as some lecturers, 63 male and 16 female of 22,7 years of age on average.



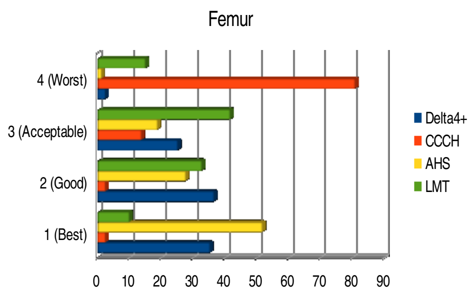


Fig. 9. Users´ Assessment for the 3D Reconstructed Veins (left) and Femur (right)

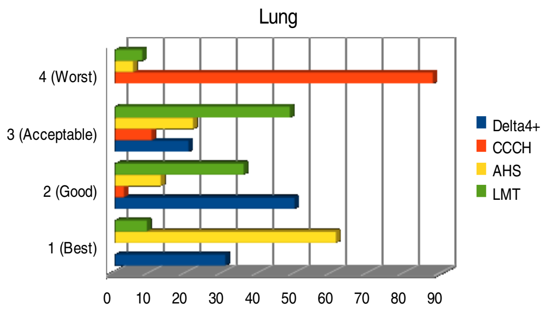
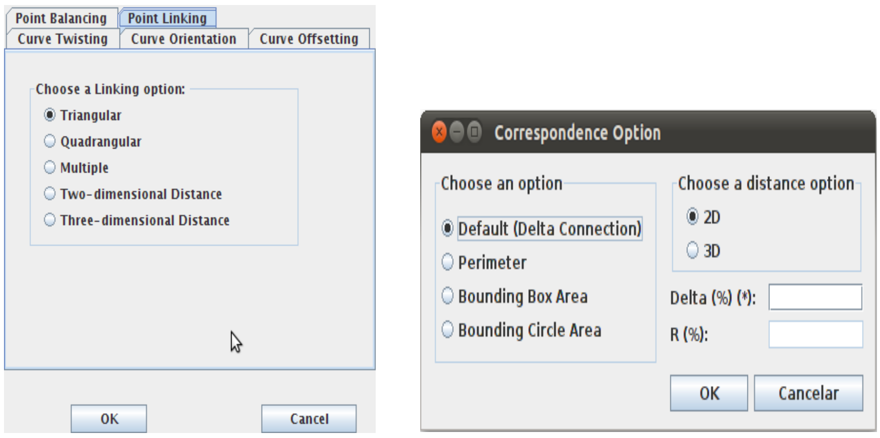


Fig. 10. Users´ Assessment for a 3D Reconstructed Lung

Figures 9 and 10 show how the subjects ranked the reconstructed models, after shuffling them back, and mean average scores where 1 means the best and 4 the worst reconstructed model. It can be noted that AHS was the user's preferred tiling approach for all models and CCCH was considered the worst model in most cases.

6.1 Using the Software

A software has been developed, called Delta Connection Plus, where users can select which solution is to be applied for every step of the tiling phase. The solutions that were implemented are those summarized in Fig. 2 but the software is modular to encourage programmers to include their own atomic contributions. The software produces X3D [13] 3D models that can be visualized in an internet browser using BS Contact [14] free plug-in. To generate a reconstructed 3D model, a heuristic approach for the Correspondence phase is mandatory. A variation of [1] was implemented but its discussion was left out of this text due to space restrictions. Fig. 11 presents two frames of the software where there are the options for selecting tiling and correspondence approaches.



Choose a Linking option:
[✓ Triangular]
Quadrangular
Multiple
Two-dimensional Distance
Three-dimensional Distance

Choose an option
[✓ Default (Delta Connection)]
Perimeter
Bounding Box Area
Bounding Circle Area

Choose a distance option
[✓ 2D]
3D

Delta (%) (*):
R (%):

Fig. 11. Choosing Solutions for the Points Linking Step and Correspondence Phase

6.2 Analysis

Tables 6, 7 and 8 summarize the results. They highlight the time, geometrical and user-centered metrics. The columns with the "↑" symbol shows the number of times the tiling approach produced the "best" result for the given criteria where best result could be related to a smaller value (like in the time criteria) or a bigger value (like in the count of user´s preference) when comparing approaches; the column with "↓" shows the number of times that the approach produced the "worst" result, and; the column with a "◊" symbol indicates the "second best" approach.

Table 6. Statistics for Best and Worst results on Geometrical Criteria

Geometrical	↑	↓	◊
AHS	2	10	4
LMT	4	3	4
CCCH	4	11	1
Delta1+		1	1
Delta2+	1		2
Delta3+	1		8
Delta4+	14		1

Table 7. Statistics for Best and Worst results on Performance Criteria

Performance	↑	↓	◊
AHS		1	
LMT			4
CCCH	4		
Delta1+		1	
Delta2+			
Delta3+			
Delta4+		3	

Table 8. Statistics for Best and Worst results from User´s Point of View

User's	↑	↓	◊
AHS	3		
LMT		1	
CCCH		2	
Delta4+			3

For instance, Delta4+ scored 14 at "↑" column in Table 6 because while analyzing the Tables 2, 3, 4 and 5, this approach happens to have produced 14 times the best values for the geometrical metrics (such as the average, media, maximum and standard deviation values for the veins model). On the other hand, CCCH approach accounted for 11 indications of worst results on those geometrical metrics. For the geometrical metrics, minimum span size was disregarded because they ended up with the same value for most of the situations.

Although a much bigger variety of models should be analyzed, the data gathered from Tables 2 to 5 were consistent enough to suggest that Delta4+ presented the best result for the majority of geometrical metrics while CCCH is the fastest approach ever. And, contrary happens too often, i.e., CCCH showed the worst geometrical results and Delta4+ showed the slower solution. It suggests that the simpler and straightforward solution (which is finding the smaller span while linking contours), although the fastest, do not end up with the desired effect (i.e., the smallest span on the whole model) but poor performance on the span-related metrics. Therefore, it can be concluded that the more heuristics taken for the steps of the tiling phase of a 3D reconstruction approach, the better the geometrical features of the model.

The performance analysis took the models with the same number of points but with different quantities of slices and contour outlines. Thus, all tests roughly start from the same magnitude of data. However, performance cannot be considered in isolation since the approach which used the shortest span as a local criterion produced the worst visual results.

All output data, the application and source code are available as GPL free software license on `www2.joinville.udesc.br/~larva/deltaplusconnection` (in Portuguese) as well as on the following public repository: `deltaconnection.googlecode.com/svn/branches/branch-des-deltaconnection 2.0.0-23032009/DeltaConnection`.

7 Conclusions

This paper compared heuristic approaches for the "tiling" phase of 3D reconstruction algorithms considering: the criteria to assess the resulting model; the constructs of the approaches and; user´s perception of the 3D reconstructed model.

Regarding the model´s assessment, we conclude that simple and straightforward heuristics (use of the shortest span) do not render a model that complies with that very same criterion for the whole model (as a global criterion). Instead, extra heuristics (such as the Points Balancing step) are needed to achieve the criterion. But, there is a clear tradeoff between performance and geometrical criteria which highlight the fact that applying extra heuristics surely impact the model´s quality (because they help avoid weird, twisted and irregular meshes) but demands extra time to be performed. Therefore, there is no best approach to be sought for all cases; rather, a mix of solutions can be tailored to meet specific time and quality criteria.

Regarding the constructs of the approaches, the "Divide to Conquer" strategy was the method used to break down tiling algorithms into five steps that comply to existing approaches. Thereafter, a software was implemented that allows a variety of algorithmic pieces to be proposed and mixed together instead of proposing a brand new (single) tiling approach. Regardless its origins (either from existing tiling approaches or new ones), we showed that there is room for improvements on every single step of the tiling phase and particular solutions can be fine tuned for particular needs. This strategy produced a solution for the tiling phase which proved to be better than existing ones on geometrical aspects but not on performance. For instance, a mix called Delta4+, bettered four out of six geometrical criteria used for comparison.

Regarding user-centered assessment we conclude that this criterion alone seems not to be a good way to evaluate a 3D reconstructed model because (a) the difficulty of observing snapshots of a 3D model which should in fact be analyzed interactively (studying the model in a 3D environment) from all viewpoints as well as inside out (which poses another challenge of how to observe inner intricacies of the model); (b) user analysis correlated only with one of geometrical criterion of quality but not with the remaining criteria; (c) each model contains a huge amount of details that could be so alike that are difficult to differentiate visually, and; (d) users tend to evaluate the overall appearance of the model compared to their own personal imagination of how that object should be instead of actually assessing the quality of the face mesh.

To summarize, we argue that span-related geometrical criteria (such as median or average) should be used as a quality criterion to compose torsion-free 3D reconstructed models and; we found that user´s visual assessment is not reliable.

As future work, it could be interesting to investigate if the overall shape of the intended object-of-study has any influence on selecting the atomic solutions on each step or, if new contributions to that specific type of object can be devised.

References

1. Anzolin, G.R., Hounsell, M.S., Silva, A.G.: Delta-Connection: A Solution for 3D Object Reconstruction. INFOCOMP Journal of Computer Science 7(2), 65–73 (2008) ISSN: 1807-4545
2. Barequet, G.: Publicly-Available Resources, `ftp://ftp.cs.technion.ac.il/pub/barequet/psdb` (accessed November 13, 2013)
3. Barrequet, G., Shapiro, D., Tal, A.: Multilevel Sensitive Reconstruction Polyhedral Surfaces from Parallel Sections. The Visual Computer 16, 116–133 (2000)
4. Chen, Y., Chen, Y., Chiang, A., Hsieh, K.: A Reliable Surface Reconstruction System in Biomedicine 86(2), 141–152 (2007), doi:10.1016/j.cmpb.2007.01.011, ISSN 0169-2607
5. Cristiansen, H.N., Serderberg, T.W.: Conversion of Complex Contour Line Definition into Polygonal Element Mosaics. Computer Graphics 12, 187–192 (1978)
6. Fuchs, H., Kedem, Z.M., Useltonm, S.: Optimal Surface Reconstruction from Planar Contours. Communications of the ACM 20(10), 693–702 (1977)
7. Keppel, E.: Approximating Complex Surfaces by Triangulation of Contour Lines. IBM J. Res. Develop. 19, 2–11 (1975)
8. Kaneda, K., Harada, K., Nakamae, E., Yasuda, M., Sato, A.G.: Reconstruction and Semi-Transparent Display Method for Observing Inner Structure of an Object Consisting of Multiple Surfaces. The Visual Computer 3, 137–144 (1987)
9. Li, Z., Ma, L., Tan, W.: Three-dimensional Object Reconstruction from Contour Lines. In: Proceedings of the ACM International Conference on Virtual Reality Continuum and its Applications, Hong Kong, pp. 319–322 (June 2006)
10. Meyers, D., Skinner, S., Sloan, K.: Surface from Contours. ACM Trans. on Graphics 11(3), 228–258 (1992)
11. Meyes, D.: Reconstruction of Surfaces From Planar Contours. Doctor of Philosophy. University of Washington (1994)
12. Sloan, K.R., Painter, J.: From Contours to Surfaces: Testbed and Initial Results. In: Proc. CHI+GI 1987, Toronto, Canada, pp. 115–120 (April 1987)
13. Web3d Consortium. X3D Frequently Asked Questions, `http://www.web3d.org` (accessed on February 15, 2012)
14. BS Contact, `http://www.bitmanagement.com` (accessed on February 15, 2012)
15. Barequet, G., Shapiro, D., Tal, A.: History Consideration in Reconstructing Plyhedral Surfaces from Parallel Slices. In: Visualization 1996, pp. 149–156 (1996)

$\sqrt{3}$-Subdivision Wavelets for Sharp Features Preservation

Hong Xiao[*], Yuan Li, Jianfeng Yu, and Jie Zhang

The Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an 710072, China
xiaohongjxr@mail.nwpu.edu.cn, {yuanli,yujf,zhangjie}@nwpu.edu.cn

Abstract. Multiresolution representation and analysis of 3D models play an important role in applications such as progressive transmission, rendering and real-time interaction of complex 3D models. Since $\sqrt{3}$ subdivision is the slowest topological refinement scheme among the triangular subdivisions, and wavelets provide a natural framework for multiresolution analysis of functions, we construct a new type of $\sqrt{3}$-subdivision wavelets using local operators. In order to maintain sharp features of 3D models, we introduce a method for sharp features identification and preservation. Subsequently, we extend the local operators to construct $\sqrt{3}$-subdivision wavelets for sharp features preservation. The experiments show that the proposed $\sqrt{3}$-subdivision wavelets can generate more levels of detail and maintain sharp features well when decomposing 3D models, compared with the other subdivision wavelets. Moreover, the computation involved in obtaining the multiresolution representation of 3D models is efficient, and linear in complexity.

Keywords: $\sqrt{3}$ subdivision, wavelet transform, 3D models, sharp features, multiresolution.

1 Introduction

Subdivision surfaces are widely employed in computer graphics and geometric modeling due to their advantages such as simplicity, strong adaptability, high efficiency and stability. The basic principle of subdivision surfaces is that, subdividing a given initial mesh recursively by certain subdivision scheme, consequently generating a series of refined meshes, and converging to the limit surface ultimately. So subdivision surfaces have superiority in multiresolution representation of 3D models. Meanwhile, wavelets provide a natural framework for multiresolution representation and analysis. It is quite natural and necessary to combine them in order to represent highly detailed 3D models. Subdivision wavelets provide an effective multiresolution representation and analysis framework for 3D models, which plays an important role in applications such as progressive transmission, rendering and real-time interaction of complex 3D models. Besides,

[*] Corresponding author.

sharp features are the key to maintain the appearance of 3D models [19], [20], especially in engineering domain. We aim to investigate a subdivision wavelets scheme with sharp features preservation in order to build a theoretical basis for delivering 3D models to the production field of complex product on mobile devices [21], [22].

A lot of subdivision schemes are proposed for triangular and quadrilateral meshes of arbitrary topology such as Loop subdivision [1], Butterfly subdivision [2], 4-8 subdivision [3], $\sqrt{3}$ subdivision [13] for triangular meshes and Doo-Sabin subdivision [4], Catmull-Clark subdivision [5], $\sqrt{2}$ subdivision [6] for quadrilateral meshes. Since $\sqrt{3}$ subdivision is the slowest refinement scheme among the current subdivision schemes for triangular meshes, we present a new subdivision wavelets construction based on $\sqrt{3}$ subdivision. However, it should be pointed out that $\sqrt{2}$ subdivision, which is a subdivision scheme for quadrilateral meshes and only doubles the number of faces in a subdivision step, refines a mesh slower than $\sqrt{3}$ subdivision does. But triangle is in a sense the easiest and best-behaved geometric primitive among point, line, triangle, quadrilateral, and polygon [16]. That is also the reason why most hardware-accelerated 3D engines choose triangle as the only truly native rendering primitive, and the mobile 3D Graphics API standard OpenGL ES only supports triangles. Based on the same consideration, we narrow the scope of this article to $\sqrt{3}$-subdivision wavelets construction.

A number of subdivision wavelets have been proposed in literature since the middle of 1990s. Lounsbery et al. [7] made the first contribution to connect wavelets and subdivision scheme to define different levels of resolution. However, Lounsbery's scheme only suits meshes having regular subdivision connectivity. Khodakovsky and his coworkers [8] improved Lounsbery's scheme so that wavelet analysis can be constructed on semi-regular meshes, i.e., meshes built by successive subdividing starting from a coarse irregular mesh. Almost all vertices in a semi-regular mesh have valence 6 (for meshes obtained based on triangular subdivision algorithm), where the valence of a vertex denotes the number of edges sharing the vertex. Unfortunately, both the schemes in [7] and [8] need solve global sparse linear systems when computing the forward wavelet transform. And the stability of their wavelet decomposition is poor, since it is observed that the numerical condition number of seven levels of wavelet-decomposition is usually below 30 [9]. Bertram [10] proposed a biorthogonal Loop-subdivision wavelets scheme. His wavelets located at the new inserted vertices are orthogonalized with respect to local scaling functions located at vertices of the corresponding triangles. The contribution of Bertram's approach is local computation of both wavelet analysis and synthesis in linear time, avoiding the solution of global sparse linear systems. Similarly, Li et al. [17] constructed unlifted Loop-subdivision wavelets based on local operators by optimizing free parameters in subdivision steps. And the Loop-subdivision wavelets were extended to support sharp features preservation. Charina and his coworker [11] presented multiresolution representation and analysis method based on subdivision wavelet tight frames. Wavelet tight frames benefited the method a lot. Firstly, a tight frame is its own dual and, therefore, the masks for decomposition and reconstruction are the same. So there is no need to solve any linear system of equations. Secondly, the decomposition over an arbitrary number of resolution levels has condition number 1, i.e., the method based on subdivision wavelet tight frames is very stable. Liang et al. [12] showed some applications of Loop-subdivision wavelet tight frames to process 3D graphics, and the experiments demonstrated that Loop-subdivision wavelet tight frames performed better than biorthogonal Loop-subdivision wavelets in 3D models compression and progressive transmission.

All the multiresolution representation and analysis schemes presented above are based on Loop subdivision. As mentioned before, $\sqrt{3}$ subdivision is the slowest refinement scheme among the current subdivision schemes for triangular meshes. Moreover, $\sqrt{3}$ subdivision was proved to be an efficient view-dependent refinement [14]. Wang et al. [15] made a meaningful attempt to construct $\sqrt{3}$-subdivision-based biorthogonal wavelets.

Benefited from the previous works, we construct $\sqrt{3}$-subdivision wavelets by using local operators. In addition, in order to keep the sharp features of 3D models, we extend the local operators to improve $\sqrt{3}$-subdivision wavelets for sharp features preservation.

2 The Proposed $\sqrt{3}$-Subdivision Wavelets

2.1 A Brief Introduction to $\sqrt{3}$ Subdivision

$\sqrt{3}$ subdivision, introduced by Kobbelt [13], is a approximating subdivision scheme for triangular meshes of arbitrary topology. Unlike Loop subdivision, which uses the normal 1-to-4 splitting operator where the mesh is refined by inserting one new vertex per edge, $\sqrt{3}$ subdivision inserts one new vertex in per triangle at its center and thus executes a 1-to-3 splitting operator in every refinement step. In short, Loop-subdivision splits edges while $\sqrt{3}$ subdivision splits triangles. One 1-to-3 splitting operator introduces three new edges connecting the new vertex to the surrounding old ones. In order to re-balance the valence of the mesh vertices, every original edge that connects two old vertices is flipped. This type of splitting has the positive effect that all newly inserted vertices have valence 6 and the valences of the old vertices does not change, as shown in Fig. 1. The subdivision masks are depicted in Fig. 2, expressed as:

$$\begin{aligned} &q = \tfrac{1}{3}\left(p_i + p_j + p_k\right), \\ &p = \left(1 - \alpha_n\right) p + \alpha_n \tfrac{1}{n} \sum_{i=0}^{n-1} p_i, \\ &\alpha_n = \tfrac{4 - 2\cos\left(\frac{2\pi}{n}\right)}{9}. \end{aligned} \tag{1}$$

$\sqrt{3}$ subdivision surface can be described by the control mesh M and corresponding control point coordinates V. A sequence of finer and finer semi-regular meshes $M^1, M^2, ..., M^\infty$ whose limit is a $\sqrt{3}$ subdivision surface will be obtained by applying the $\sqrt{3}$ subdivision operator recursively on an irregular base mesh M^0. v^i denotes the vertices in M^i, and $\varepsilon^i = v^{i+1} - v^i$ is a set of vertices

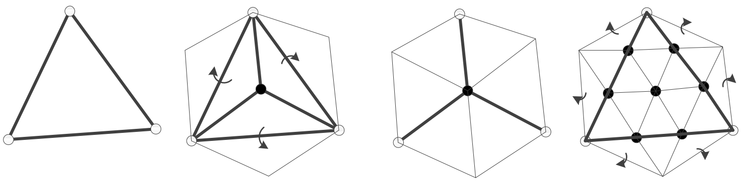


Fig. 1. Two split operations of $\sqrt{3}$ subdivision

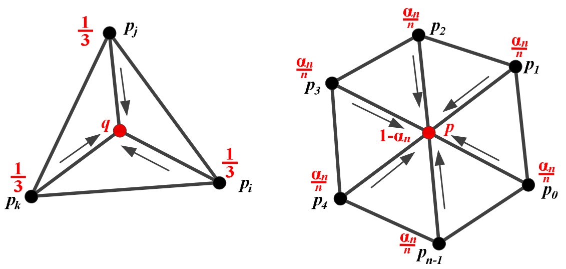


Fig. 2. Masks for $\sqrt{3}$ subdivision

corresponding to faces at level i. V^i and F^i denote the point coordinates sets associated with v^i and ε^i respectively. The substance of multiresolution analysis for 3D model is decomposing the high-resolution model $\left\{M^{i+1}, V^{i+1}\right\}$ into low-resolution representation $\left\{M^i, V^i\right\}$ and corresponding wavelet coefficients F^i [7], [17], i.e., $V^i = A \cdot V^{i+1}$, $F^i = B \cdot V^{i+1}$, where A and B denote the low-pass and high-pass filters respectively, collectively called analysis filters. The wavelet reconstruction process is the inverse of the decomposition process. The refining filter P and perturbing filter Q, called synthesis filters, are constructed in order to recover the original model from the low-resolution version and the wavelet coefficients, i.e., $V^{i+1} = P \cdot V^i + Q \cdot F^i$.

In this paper, we decompose the $\sqrt{3}$ subdivision rules into a series of reversible local operators to construct the proposed $\sqrt{3}$ subdivision wavelets. These local operators consist of the $\sqrt{3}$ subdivision wavelets reconstruction rules. Then the $\sqrt{3}$ subdivision wavelets decomposition rules can be established by inverting the $\sqrt{3}$ subdivision wavelets reconstruction rules and arranging them in the reverse order [15], [17].

2.2 Local Operators of $\sqrt{3}$ Subdivision

After analyzing the $\sqrt{3}$ subdivision rules (see Fig. 2 and Equation 1), we decompose them into the following two local operations, as shown in Fig. 3.

So the $\sqrt{3}$ subdivision rules can be expressed as:

$$\begin{array}{l} q \leftarrow \frac{1}{3}\left(p \rightarrow q\right), \\ p \leftarrow c_1 p + c_2\left(q \rightarrow p\right) \end{array} \tag{2}$$

where $c_1 = \cos\left(\frac{2\pi}{n}\right) - 1$ and $c_2 = \frac{1-c_1}{3n}$.

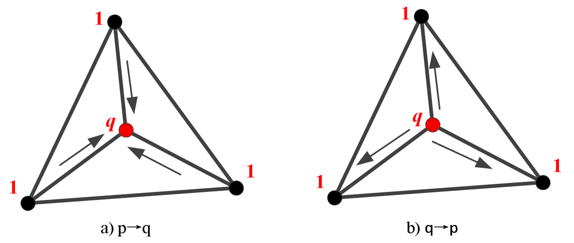


Fig. 3. Masks for local operators of $\sqrt{3}$ subdivision

2.3 $\sqrt{3}$-Subdivision Wavelets Reconstruction Rules

With the purpose of making Equation 2 reversible, an additional lifting operator is added prior to the $\sqrt{3}$ subdivision rules. So that the $\sqrt{3}$-subdivision wavelets reconstruction rules are:

$$\begin{array}{l} p \leftarrow p + d_1\,(q \rightarrow p) \\ q \leftarrow d_2 q + \frac{1}{3}\,(p \rightarrow q) \\ p \leftarrow c_1 p + c_2\,(q \rightarrow p) \end{array} \tag{3}$$

where d_1 and $d_2 (d_2 \neq 0)$ are the newly introduced parameters.

2.4 $\sqrt{3}$-Subdivision Wavelets Decomposition Rules

It is convenient to construct the $\sqrt{3}$-subdivision wavelets decomposition rules as the inverse of Equation 3, that is

$$\begin{array}{l} p \leftarrow \frac{[p - c_2 (q \rightarrow p)]}{c_1} \\ q \leftarrow \frac{q - \frac{1}{3}(p \rightarrow q)}{d_2} \\ p \leftarrow p - d_1\,(q \rightarrow p) \end{array} \tag{4}$$

2.5 Parameters Optimizing for Orthogonalization

As we can see, Equations 3 and 4 constitute the multiresolution synthesis and analysis framework of $\sqrt{3}$-subdivision wavelets. In order to eliminate the data correlation, the wavelets functions located at the new inserted vertices are orthogonalized with scaling functions located at vertices of the corresponding triangles as much as possible [7], [17], so that we can get the optimal values of parameters d_1 and d_2, i.e.,

$$\forall f \in \varepsilon^i, \min_{d_1, d_2} \sum_{v \in v^i} \frac{\left\langle \psi_f^i, \varphi_v^i \right\rangle^2}{\left\langle \psi_f^i, \psi_f^i \right\rangle \langle \varphi_v^i, \varphi_v^i \rangle} \tag{5}$$

Solving the extreme-value problem of Equation 5 needs calculating three inner products first, i.e., inner product of wavelets functions and scaling functions

$\left\langle \psi_f^i, \varphi_v^i \right\rangle$, inner product of wavelets functions $\left\langle \psi_f^i, \psi_f^i \right\rangle$, as well as inner product of scaling functions $\left\langle \varphi_v^i, \varphi_v^i \right\rangle$.

According to the multi-scale relationship of wavelet transform, low-resolution scaling function φ^i and wavelets function ψ^i can be expressed as a linear combination of high-resolution scaling function φ^{i+1}:

$$\varphi_{v \in v^i}^i = \sum_{v' \in v^{i+1}} h_{v,v'} \varphi_{v'}^{i+1} \tag{6}$$

$$\psi_{f \in \varepsilon^i}^i = \sum_{v' \in v^{i+1}} g_{f,v'} \varphi_{v'}^{i+1} \tag{7}$$

where $h_{v,v'}$ is determined by $\sqrt{3}$ subdivision rules, and $g_{f,v'}$ depends on the specific wavelets format.

The preparation for calculating the three inner products in Equation 5 is the calculation of inner product of high-resolution scaling functions $\left\langle \varphi_{v'}^{i+1}, \varphi_{v''}^{i+1} \right\rangle$. Combining with Equation 7, the inner product of wavelets functions and high-resolution scaling functions is shown as followed:

$$\left\langle \varphi_{v'}^{i+1}, \psi_f^i \right\rangle = \sum_{v'' \in v^{i+1}} g_{f,v''} \left\langle \varphi_{v'}^{i+1}, \varphi_{v''}^{i+1} \right\rangle \tag{8}$$

We can acquire the inner product of low-resolution scaling functions and wavelets functions according to Equations 6 and 8, that is

$$\left\langle \varphi_v^i, \psi_f^i \right\rangle = \sum_{v' \in v^{i+1}} h_{v,v'} \left\langle \varphi_{v'}^{i+1}, \psi_f^i \right\rangle \tag{9}$$

In a similar way, the inner product of wavelets functions is

$$\left\langle \psi_f^i, \psi_f^i \right\rangle = \sum_{v' \in v^{i+1}} g_{f,v'} \left\langle \varphi_{v'}^{i+1}, \psi_f^i \right\rangle \tag{10}$$

By substituting the inner product of scaling functions $\left\langle \varphi_v^i, \varphi_v^i \right\rangle$, Equations 9 and 10 into Equation 5, it will be expressed as a function of parameters d_1 and d_2. So, Equation 5 is transformed into an extreme-value problem about parameters d_1 and d_2. The details of the optimization procedure are cumbersome and tedious. For the sake of brevity and compactness of this paper, we do not present detailed description of the optimization procedure and one can refer to Reference [17] for the specific calculation process.

3 Extension of $\sqrt{3}$-Subdivision Wavelets for Sharp Features Preservation

3.1 Extension of $\sqrt{3}$ Subdivision Rules

Some features in 3D models play a more important role than the other ones for the shape and topology of the models, such as the sharp features. As stated earlier, $\sqrt{3}$ subdivision scheme has a lot of advantages. However, it emphasizes on the smoothness too much whereas the sharp features would be missed [18]. Therefore, it is important to identify and keep these sharp features, and extend the $\sqrt{3}$ subdivision rules to support them. Yu et al. [18] compared the normal angle of two triangles sharing one edge with a given threshold θ. If the normal angle was larger than θ, the edge would be tagged as *Sharp*, else be tagged as *Smooth*. And a vertex would be tagged according to its sharp degree s, namely the number of incident sharp edges. A smooth vertex is the one with sharp degree $s = 0$; a dart vertex has $s = 1$; crease vertex has $s = 2$; and corner vertex is the one has sharp degree $s > 2$. In this paper, we will adopt Yu's method which has been proven to be effective, and mark all boundary edges of the mesh as sharp.

After the sharp features identification, we need to improve the $\sqrt{3}$ subdivision rules to support sharp features preservation. We will insert two edge vertices on the sharp edges (double lines in Fig. 4) at the even refinement step, update and connect them with the other vertices in the mesh. In this way, we will acquire the same 1-to-9 result as two original $\sqrt{3}$ subdivision operators will do and keep the sharp edges in the meantime, as shown in Fig. 1 and Fig. 4. The masks for $\sqrt{3}$ subdivision preserving sharp features is demonstrated in Fig. 5.

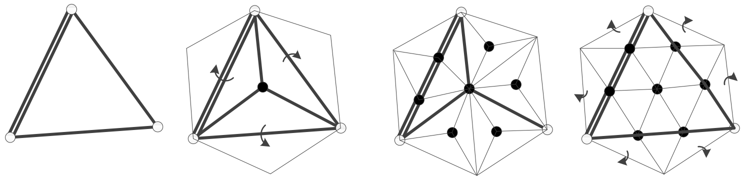


Fig. 4. Two split operations of $\sqrt{3}$ subdivision for sharp features preservation

According to the $\sqrt{3}$ subdivision rules for sharp features preservation, we extend the local operators in Section 2.2, as shown in Fig. 6.

So the extended $\sqrt{3}$ subdivision rules can be expressed as follow:

$$\begin{aligned}
&1)\, q \leftarrow \tfrac{1}{3}\left(p \to q_{0/3}\right),\ \textit{non or all sharp edges}\\
&q \leftarrow \tfrac{1}{5}\left(p \to q_1\right),\ \textit{one sharp edge}\\
&q \leftarrow \tfrac{1}{4}\left(p \to q_2\right),\ \textit{two sharp edges}\\
&2)\, p \leftarrow c_1 p + c_2\left(q \to p\right),\ \textit{smooth or dart vertex}\\
&p \leftarrow \tfrac{7}{9}p + \tfrac{1}{9}\left(p \to p^{odd}_{crease}\right),\ \textit{vertex of a crease at the odd refinement step}\\
&p \leftarrow \lambda p + \tfrac{1}{3}\left(p \to p^{odd}_{\text{edge}}\right),\ \textit{edge point of a crease at the odd refinement step}\\
&p \leftarrow \tfrac{1}{3}p + \tfrac{1}{3}\left(p \to p^{even}_{\text{crease}}\right),\ \textit{vertex of a crease at the even refinement step}\\
&p \leftarrow \tfrac{1}{3}p + \tfrac{1}{3}\left(p \to p^{even}_{edge}\right),\ \textit{edge point of a crease at the even refinement step}\\
&p \leftarrow p,\ \textit{corner vertex}
\end{aligned} \tag{11}$$

where λ is a small nonzero, whose function is to make the steps in Equation 11 reversible.

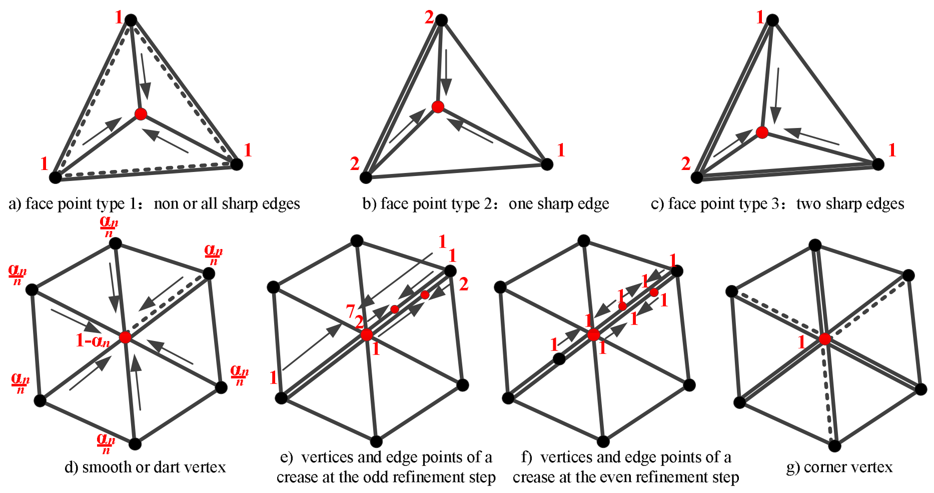

a) face point type 1：non or all sharp edges
b) face point type 2：one sharp edge
c) face point type 3：two sharp edges
d) smooth or dart vertex
$\frac{\alpha_n}{n}$
$1-\alpha_n$
e) vertices and edge points of a crease at the odd refinement step
f) vertices and edge points of a crease at the even refinement step
g) corner vertex

Fig. 5. Masks for $\sqrt{3}$ subdivision preserving sharp features

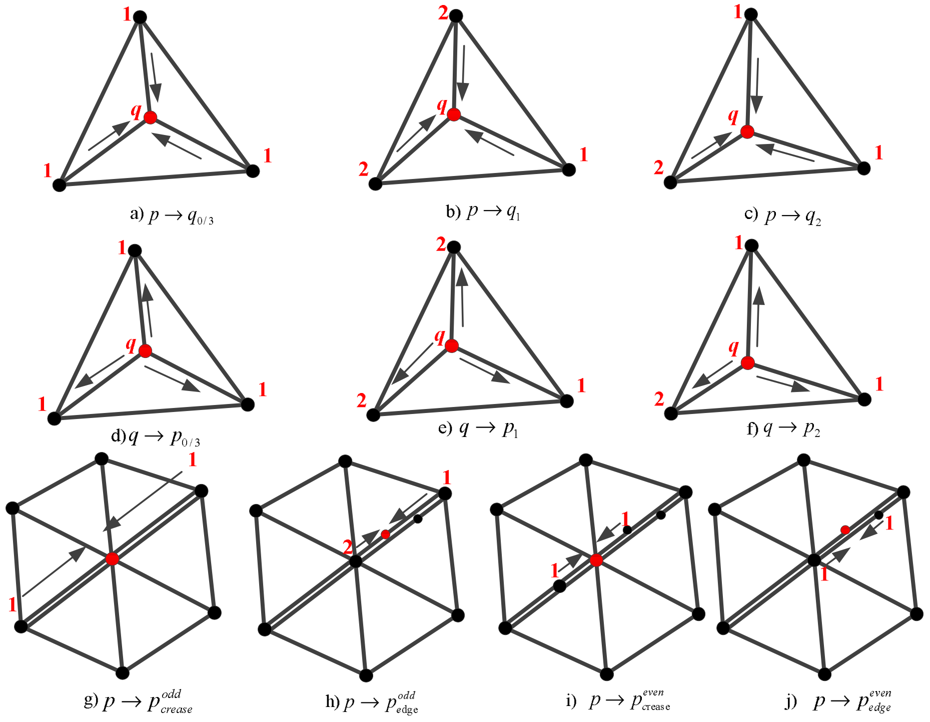

a) $p \rightarrow q_{0/3}$
b) $p \rightarrow q_1$
c) $p \rightarrow q_2$
d) $q \rightarrow p_{0/3}$
e) $q \rightarrow p_1$
f) $q \rightarrow p_2$
g) $p \rightarrow p^{odd}_{crease}$
h) $p \rightarrow p^{odd}_{edge}$
i) $p \rightarrow p^{even}_{crease}$
j) $p \rightarrow p^{even}_{edge}$

Fig. 6. Masks for the extended local operators of $\sqrt{3}$ subdivision

3.2 $\sqrt{3}$-Subdivision Wavelets Reconstruction Rules for Sharp Features Preservation

Based on Equation 11, we will easily acquire the extended $\sqrt{3}$-subdivision wavelets reconstruction rules for sharp features preservation by adding an additional lifting operator prior to this equation, i.e.

$$\begin{array}{l}
1)\, p \leftarrow p + d_1\left(q \to p_{0/3}\right),\ \textit{non or all sharp edges}\\
p \leftarrow p + d_1\left(q \to p_1\right),\ \textit{one sharp edge}\\
p \leftarrow p + d_1\left(q \to p_2\right),\ \textit{two sharp edges}\\
2)\, q \leftarrow d_2 q + \frac{1}{3}\left(p \to q_{0/3}\right),\ \textit{non or all sharp edges}\\
q \leftarrow d_2 q + \frac{1}{5}\left(p \to q_1\right),\ \textit{one sharp edge}\\
q \leftarrow d_2 q + \frac{1}{4}\left(p \to q_2\right),\ \textit{two sharp edges}\\
3)\, p \leftarrow c_1 p + c_2\left(q \to p_{0/3}\right),\ \textit{smooth or dart vertex}\\
p \leftarrow \frac{7}{9}p + \frac{1}{9}\left(p \to p_{crease}^{odd}\right),\ \textit{vertex of a crease at the odd refinement step}\\
p \leftarrow \lambda p + \frac{1}{3}\left(p \to p_{\text{edge}}^{odd}\right),\ \textit{edge point of a crease at the odd refinement step}\\
p \leftarrow \frac{1}{3}p + \frac{1}{3}\left(p \to p_{\text{crease}}^{even}\right),\ \textit{vertex of a crease at the even refinement step}\\
p \leftarrow \frac{1}{3}p + \frac{1}{3}\left(p \to p_{edge}^{even}\right),\ \textit{edge point of a crease at the even refinement step}\\
p \leftarrow p,\ \textit{corner vertex}
\end{array} \tag{12}$$

3.3 $\sqrt{3}$-Subdivision Wavelets Decomposition Rules for Sharp Features Preservation

The subdivision wavelets decompostion rules can be constructed by inverting each step of the corresponding subdivision wavelets reconstruction rules and arranging them in the reverse order. So, by inverting Equation 12 and arranging them in the reverse order, the extended $\sqrt{3}$-subdivision wavelets decomposition rules for sharp features preservation are:

$$\begin{array}{l}
1)\, p \leftarrow \frac{\left[p - c_2\left(q \to p_{0/3}\right)\right]}{c_1},\ \textit{smooth or dart vertex}\\
p \leftarrow \frac{9p - \left(p \to p_{crease}^{odd}\right)}{7},\ \textit{vertex of a crease at the odd refinement step}\\
p \leftarrow \frac{p - \frac{1}{3}\left(p \to p_{\text{edge}}^{odd}\right)}{\lambda},\ \textit{edge point of a crease at the odd refinement step}\\
p \leftarrow 3p - \left(p \to p_{\text{crease}}^{even}\right),\ \textit{vertex of a crease at the even refinement step}\\
p \leftarrow 3p - \left(p \to p_{edge}^{even}\right),\ \textit{edge point of a crease at the even refinement step}\\
p \leftarrow p,\ \textit{corner vertex}\\
2)\, q \leftarrow \frac{q - \frac{1}{3}\left(p \to q_{0/3}\right)}{d_2},\ \textit{non or all sharp edges}\\
q \leftarrow \frac{q - \frac{1}{5}\left(p \to q_1\right)}{d_2},\ \textit{one sharp edge}\\
q \leftarrow \frac{q - \frac{1}{4}\left(p \to q_2\right)}{d_2},\ \textit{two sharp edges}\\
3)\, p \leftarrow p - d_1\left(q \to p_{0/3}\right),\ \textit{non or all sharp edges}\\
p \leftarrow p - d_1\left(q \to p_1\right),\ \textit{one sharp edge}\\
p \leftarrow p - d_1\left(q \to p_2\right),\ \textit{two sharp edges}
\end{array} \tag{13}$$

4 Results

To demonstrate the performance of the proposed $\sqrt{3}$-subdivision wavelets, we have coded our algorithm in C++ based on the open source library Visualization Toolkit (VTK). The test is carried out on a PC equipped with a 3.3 GHz Intel i3 CPU and 4G RAM. In Fig. 7, a Venus base mesh is subdivided six times by the $\sqrt{3}$ subdivision rules down to the sixth level and it is regarded as the input model. Fig. 7a demonstrates the proposed wavelet analysis of Venus from level 6 to level 0 ($\theta = 130°$), while Fig. 7b is the decomposition result of Venus from level 6 to level 0 using the Loop-subdivision wavelets and Fig. 7c shows the decomposition result based on the general $\sqrt{3}$-subdivision wavelets [15]. Apparently, compared with Fig. 7b, Fig. 7a keeps more details of the mesh model, i.e., the multiresolution representation of 3D models will generate more levels of detail based on $\sqrt{3}$-subdivision wavelets than Loop-subdivision wavelets. Meanwhile, Fig. 7a indicates that the proposed $\sqrt{3}$-subdivision wavelets can preserve the sharp features other than just overemphasize the smoothness of the decomposition result [10], [15].

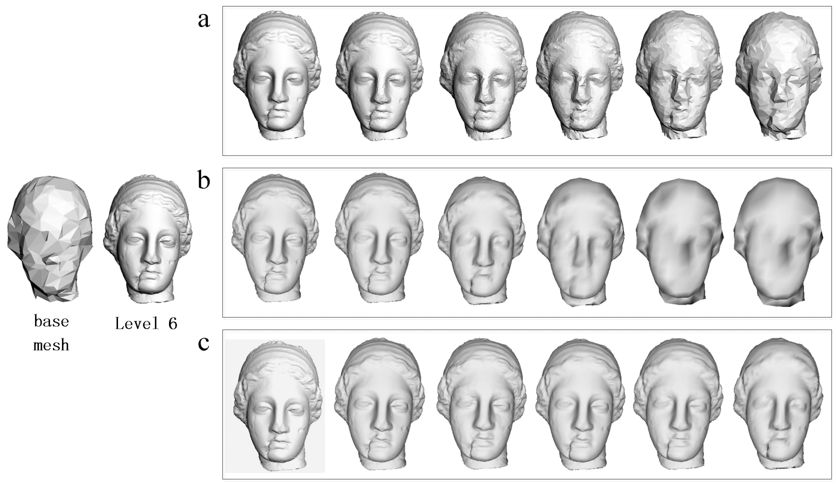


Fig. 7. Venus base mesh and the decomposition result from level 6 to level 0 using (a) the proposed sharp features preservation $\sqrt{3}$-subdivision wavelets, (b) Loop-subdivision wavelets and (c) the general $\sqrt{3}$-subdivision wavelets respectively

The low levels of model in Fig. 7a are a little rough, though. It would be necessary to analyze impacts of threshold θ on the sharp features preservation and smoothness of the decomposition result. In Fig. 8, the Horse model is decomposed from level 6 to level 0 with different threshold value. As we can see, the smaller the value of threshold θ is, the more sharp features of the model can be preserved. And the decomposition result of the model would become smoother when the value of threshold θ increases. More examples are demonstrated in Figs. 9 and 10. In Fig. 9, the decomposition result of the Skull model (its sharp features are marked green) at levels 5, 3 and 1 using the proposed $\sqrt{3}$-subdivision wavelets is shown in Fig. 9a. Correspondingly, the decomposition result of the Skull model at the same levels using the general $\sqrt{3}$-subdivision wavelets is presented in Fig. 9b. As we can see, the details of the model decrease as the level goes down, but the sharp features are maintained better in Fig. 9a. Similarly, in Fig. 10, the details attached on the Fandisk model fade gradually, while the sharp feature are preserved well.

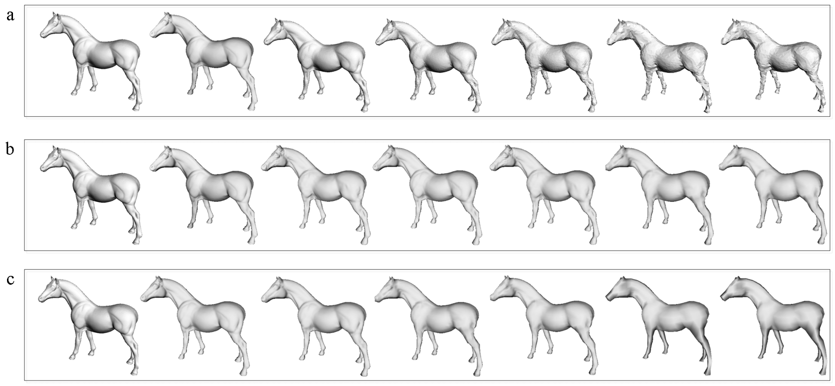


Fig. 8. The decomposition result of Horse model: (a) $\theta = 130°$; (b) $\theta = 145°$; (c) $\theta = 160°$

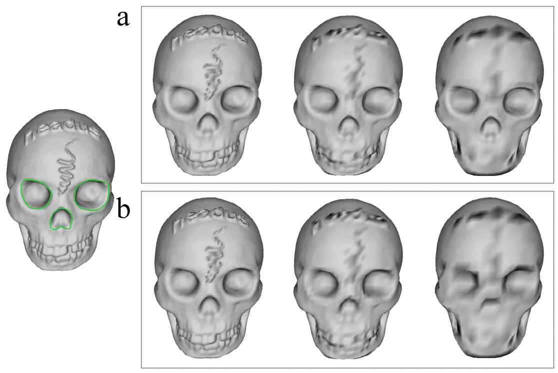


Fig. 9. The decomposition result of the Skull model at levels 5, 3 and 1 using (a) the proposed $\sqrt{3}$-subdivision wavelets, (b) the general $\sqrt{3}$-subdivision wavelets

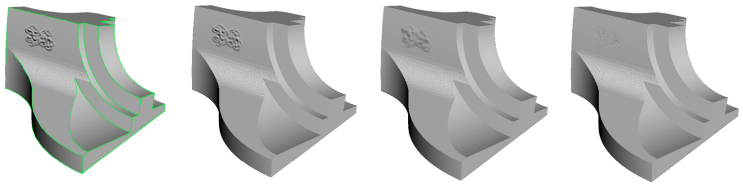


Fig. 10. The decomposition result of Fandisk model

When the scale of 3D models increases, the computational complexity of wavelet transform becomes critical [10], [15], and it is usually the criteria that whether a wavelet transform method is practical or not. Table 1 lists the computation times required for each level of wavelet decomposition of Venus and the corresponding number of triangles, as shown in Fig. 11 left. Our wavelet transform is a little more time consuming than the general $\sqrt{3}$-subdivision wavelets, but it is acceptable. Fig. 11 right shows the ratio of the computation time for reconstructing Venus at each level T_r^i to the corresponding number of triangles N_f^i, i.e. ${T_r^i}/{N_f^i}$. It is obviously that ${T_r^i}/{N_f^i}$ is approaching a constant as the level goes up. So our $\sqrt{3}$-subdivision wavelets scheme has a linear computational complexity.

Table 1. The computation times required for each level of wavelet decomposition of Venus and the corresponding number of triangles, where T_1 and T_2 denotes the time in decomposition using the general $\sqrt{3}$-subdivision wavelets and the proposed $\sqrt{3}$-subdivision wavelets respectively

level	triangle #	T_1/sec.	T_2/sec.	$\frac{T_2-T_1}{T_1} \times 100\%$
6	268272	0.194409	0.222015	14.2%
5	89424	0.075206	0.082200	9.3%
4	29808	0.034503	0.036435	5.6%
3	9936	0.012768	0.013202	3.4%
2	3312	0.006850	0.007035	2.7%
1	1104	0.003262	0.003298	1.1%
0	368	-	-	-

The computation times required for each level of wavelet decomposition of Horse with different value of threshold θ are listed in Table 2, as shown in Fig. 12. Since fewer edges of the model would be marked as sharp edge when the value of threshold θ goes up, correspondingly, much fewer sharp features preservation operation are needed and decomposing the model is less time consuming.

Progressive transmission of 3D model is an important application of the wavelet analysis and synthesis. Fig. 13 shows the progressive transmission of Dinosaur and Shell models from level 0 to level 6 ($\theta = 145°$). The computation

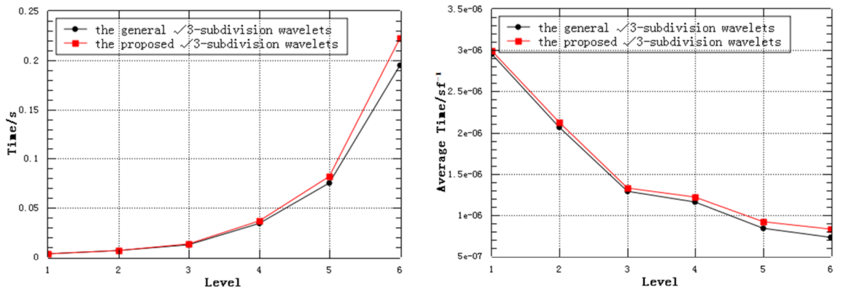


Fig. 11. Left: The computation times required for each level of wavelet decomposition of Venus; Right: the average computation times for reconstructing Venus at each level

Table 2. The computation times required for each level of wavelet decomposition of Horse with different value of threshold θ and the corresponding number of triangles

level	triangle #	$T_{\theta=130^\circ}$/s	$T_{\theta=145^\circ}$/s	$T_{\theta=160^\circ}$/s
6	242028	0.167461	0.154627	0.140954
5	80676	0.080654	0.074749	0.069599
4	26892	0.036377	0.034125	0.032133
3	8964	0.014225	0.013509	0.012829
2	2988	0.005510	0.005263	0.005105
1	996	0.004347	0.004208	0.004093
0	332	-	-	-

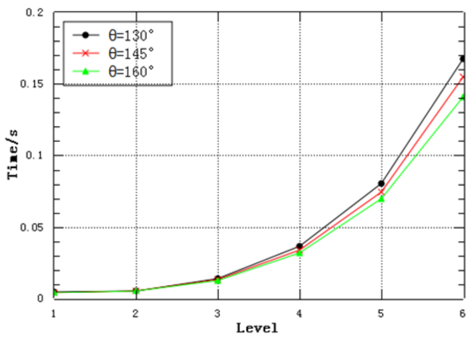


Fig. 12. The computation times required for each level of wavelet decomposition of Horse

time increases from 0.000970 to 0.097426 second when reconstructing the Dinosaur model from level 0 (including 154 triangles) to level 6 (including 112266 triangles). It costs 0.003383 second to reconstructing the Shell model at level 0 (including 537 triangles) and 0.225293 second at level 6 (including 391473 triangles). These experiments show the efficiency of the proposed wavelet transforms.

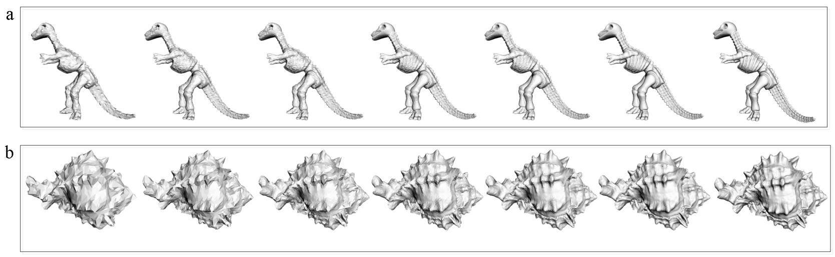


Fig. 13. Progressive transmission of Dinosaur and Shell models from level 0 to level 6

5 Summary

We present a new $\sqrt{3}$-subdivision wavelets construction using local operators. Sharp features identification and preservation method is proposed in order to extend the local operators to construct $\sqrt{3}$-subdivision wavelets for sharp features preservation. The experiments demonstrate that the multiresolution representation of 3D models will generate more levels of detail based on $\sqrt{3}$-subdivision wavelets than Loop-subdivision wavelets. Although the multiresolution decomposition results of 3D models are a little rougher than the previous wavelets, much more sharp features are preserved. Meanwhile our wavelets scheme has a linear computational complexity so that it is practical in applications such as multiresolution representation, progressive transmission, and multiresolution editing and rendering of 3D models.

Since sharp features identification are needed before multiresolution analysis of 3D models, it will take a little more time to process the models based on our wavelets scheme than the others. How to identify the sharp features of 3D models efficiently is important and it is also one of our future research directions. The threshold θ determines the two characteristics of the decomposition result of 3D model, which are the smoothness and sharp features preservation level. Choosing an appropriate threshold θ for the corresponding model to balance these two characteristics needs further research.

Acknowledgments. The work is supported by Basic Research Foundation of Northwestern Polytechnical University (Grant No. JC201137), Aeronautical Science Foundation of China (Grant No. 2009ZE53056), National High-Tech Research & Development Program of China (Grant No. 2009AA044601). And we want to express our sincere gratitude to the Caltech Multiresolution Modeling Group for the models and Dr. S Valette for his code of subdivision wavelets.

References

1. Loop, C.T.: Smooth subdivision surfaces based on triangles, Dissertation for the Master Degree. University of Utah, Utah (1987)
2. Zorin, D., Schroder, P., Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. In: Proceedings of ACM SIGGRAPH 1996, pp. 189–192. ACM Press, New York (1996)
3. Velho, L., Zorin, D.: 4-8 subdivision. Comput. Aided Geom. Design. 18, 397–427 (2001)
4. Doo, D., Sabin, M.: Behavior of recursive division surfaces near extraordinary points. Comput. Aided Design. 10, 356–360 (1978)
5. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Comput. Aided Design. 10, 350–355 (1978)
6. Li, G., Ma, W., Bao, H.: $\sqrt{2}$ subdivision for quadrilateral meshes. Visual Comput. 20, 180–198 (2004a)
7. Lounsbery, M., DeRose, T., Warren, J.: Multiresolution analysis for surfaces of arbitrary topological types. ACM Transactions on Graphics 16, 34–73 (1997)
8. Khodakovsky, A., Schroder, P., Sweldens, W.: Progressive geometry compression. In: Proceedings of SIGGRAPH 2000, pp. 271–278. ACM Press, New York (2000)
9. Charina, M., Stockler, J.: Tight wavelet frames for subdivision. Journal of Computational and Applied Mathematics 221, 293–301 (2008)
10. Bertram, M.: Biorthogonal Loop-subdivision Wavelets. Computing 72, 29–39 (2004)
11. Charina, M., Stockler, J.: Tight wavelet frames for irregular multiresolution analysis. Appl. Comput. Harmon. Anal. 25, 98–113 (2008)
12. Liang, X.Z., Xue, Y.H., Li, Q.: Some applications of Loop-subdivision wavelet tight frames to the processing of 3D graphics. Visual Computer 27, 35–43 (2011)
13. Kobbelt, L.: $\sqrt{3}$-Subdivision. In: Proceedings of SIGGRAPH 2000, pp. 103–112. ACM Press, New York (2000)
14. Alliez, P., Laurent, N., Sanson, H., et al.: Efficient View-Dependent Refinement of 3D Meshes using $\sqrt{3}$-Subdivision. Visual Computer 19, 205–221 (2003)
15. Wang, H.W., Qin, K.H., Sun, H.Q.: $\sqrt{3}$-Subdivision-Based Biorthogonal Wavelets. IEEE Transactions on Visualization and Computer Graphics 13, 914–924 (2007)
16. Pulli, K., Aarnio, T., Miettinen, V., et al.: Mobile 3D Graphics with OpenGL ES and M3G. Morgan Kaufmann, Burlington (2007)
17. Li, D.G., Qin, K.H., Sun, H.Q.: Unlifted Loop Subdivision Wavelets. In: 12th Pacific Conference on Computer Graphics and Applications, pp. 25–33. IEEE Press, New York (2004)
18. Yu, R.G., Liu, Z.Y., Liu, Y.: A Local $\sqrt{3}$-Subdivision Algorithm preserving Sharp Features. In: ICARCV 2006 9th International Conference on Control, Automation, Robotics and Vision, pp. 1–6. IEEE Press, New York (2006)
19. Shi, Z., Luo, H., Niu, X.: Saliency-based structural degradation evaluation of 3D mesh simplification. IEICE Electron. Expr. 8, 161–167 (2011)
20. Yu, Z., Holst, M.J., McCammon, J.A.: Feature-preserving adaptive mesh generation for molecular shape modeling and simulation. J. Mol. Graph. Model. 26, 1370–1380 (2008)
21. Chen, Y., Kamara, J.M.: A framework for using mobile computing for information management on construction sites. Automat. Constr. 20, 776–788 (2011)
22. Sun, X., Yang, B., Attene, M., Li, Q., Jiang, S.: Automated abstraction of building models for 3D navigation on mobile devices. In: Geoinformatics, 2011 19th International Conference on, pp. 1–6. IEEE Press, New York (2011)

BetaMDGP: Protein Structure Determination Algorithm Based on the Beta-complex

Jeongyeon Seo[1], Jae-Kwan Kim[2], Joonghyun Ryu[2], Carlile Lavor[3], Antonio Mucherino[4], and Deok-Soo Kim[1,2,⋆]

[1] Department of Industrial Engineering, Hanyang University
17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791, South Korea
jyseo@voronoi.hanyang.ac.kr, dskim@hanyang.ac.kr

[2] Voronoi Diagram Research Center, Hanyang University
17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791, South Korea
{jkkim,jhryu}@voronoi.hanyang.ac.kr, dskim@hanyang.ac.kr

[3] Dept. of Applied Math. (IMECC-UNICAMP), University of Campinas
13081-970, Campinas - SP, Brazil
clavor@ime.unicamp.br

[4] IRISA, University of Rennes I, France
antonio.mucherino@irisa.fr

Abstract. The molecular distance geometry problem (MDGP) is a fundamental problem in determining molecular structures from the NMR data. We present a heuristic algorithm, the BetaMDGP, which outperforms existing algorithms for solving the MDGP. The BetaMDGP algorithm is based on the beta-complex, which is a geometric construct extracted from the quasi-triangulation derived from the Voronoi diagram of atoms. Starting with an initial tetrahedron defined by the centers of four closely located atoms, the BetaMDGP determines a molecular structure by adding one shell of atoms around the currently determined substructure using the beta-complex. The proposed algorithm has been entirely implemented and tested with atomic arrangements stored in an NMR format created from PDB files. Experimental results are also provided to show the powerful capability of the proposed algorithm.

Keywords: Protein structure determination, Molecular Distance Geometry Problem, Voronoi Diagram, Quasi-triangulation, Beta-complex.

1 Introduction

One of the key challenges for understanding a protein function is understanding its structure as it is the determinant of molecular function [1]. There are two main experimental methods to determine protein structures: NMR spectroscopy [2] and X-ray crystallography [3]. Given an NMR spectroscopy file that defines the interatomic distances for some pairs of atoms, usually between the hydrogen atoms in a molecule, determination of the optimal assignment of coordinates that

⋆ Corresponding author.

satisfies the inter-distance constraints is required. The determinant of molecular structure from NMR spectroscopy is studied by solving the distance geometry problem (DGP), which is a well-known mathematical problem. The DGP is to find an embedding of a weighted undirected graph $G = (V, E, d)$ in an arbitrary dimensional space [4–6]. Each vertex $v \in V$ corresponds to a point x_v in space, and there is an edge between the two vertices if and only if their relative distance is known. The length d of an edge is its weight. Formally, the DGP is the problem of finding the location of x such that $u, v \in V$, $\forall(u, v) \in E$, and $|| \, x_u - x_v \, || = d_{u,v}$, where $d_{u,v}$ is the distance between u and v. Hence, the DGP is called a constraint satisfaction problem from a mathematical point of view because a set of coordinates must be found to satisfy the constraints. The DGP can be solved in polynomial time if the complete set of the exact distances is available [7] but is NP-hard for a general sparse set of distances even in three-dimensional space [8]. In other words, it is very difficult to correctly solve for the general setting of NMR spectroscopy in practice because it contains only a subset of the complete graph between hydrogen atoms.

We are interested in a particular class of the DGP called the molecular distance geometry problem (MDGP) arising in biology where the vertices of G represent the atom centers of a molecule. The aim of the MDGP is to identify the three-dimensional molecular conformation in three dimensional space using the Euclidean distance. The MDGP is of crucial importance for biomedical problems because a molecular function is primarily determined by its structure. While X-ray crystallography produces the absolute coordinates of the atom locations, the NMR produces the relative distance information among the atoms, usually within 5 Å [2]. Hence, the MDGP is the core problem for NMR technology.

Let x_i be the coordinate of the atom i and D the given set of the distance $d_{i,j}$ between the atom i and the atom j, $i \neq j$. The problem is to find x_i, $i = 1, ..., n$ such that

$$||x_i - x_j|| = d_{i,j}, \qquad \forall d_{i,j} \in D. \tag{1}$$

The most common approach to the MDGP is to formulate the problem as a continuous optimization problem [9–13].

$$Min. \sum_{(i,j) \in D} (||x_i - x_j||^2 - d_{i,j}^2)^2 \tag{2}$$

In real NMR files, the distances are given with the lower and the upper bounds [14]. The MDGP with the lower and the upper bounds is to find a set of positions $x_1, ..., x_n$ in the three-dimensional space such that

$$l_{i,j} \leq ||x_i - x_j|| \leq u_{i,j}, \qquad \forall d_{i,j} \in D \tag{3}$$

where l_{ij} and u_{ij} are the lower and the upper bounds on the distances, respectively.

The standard formulation by Crippen and Havel [5] is to solve the following minimization problem:

$$Min. \sum_{(i,j) \in D} p_{i,j}(x) \tag{4}$$

$$p_{i,j}(x) = Min^2\{\frac{||x_i - x_j||^2 - l_{i,j}^2}{l_{i,j}^2}, 0\} + Max^2\{\frac{||x_i - x_j||^2 - u_{i,j}^2}{u_{i,j}^2}, 0\} \quad (5)$$

Crippen and Havel applied the MDGP to protein modeling [5, 15, 7]. The MDGP has been studied by many groups: the embedding algorithm approach by Crippen and Havel [5, 15], the graph reduction approach by Hendrickson [16, 17], the approaches based on the global optimization method by Moré and Wu [9, 10] and An and Tao [11, 12], and the geometric build-up algorithm by Dong, Wu, Sit, and Yuan [18–21]. In particular, the embedding algorithm by Crippen and Havel has been adopted in NMR modeling through programs such as CNS, XPLOR, and XPLOR-NIH [22, 23]. Under certain assumptions, the problem can be formulated as a combinatorial optimization problem, called the discretizable MDGP (DMDGP) [24]. While the NP-hardness of the problem is unavoidable [24], the Branch and Prune (BP) algorithm solves the DMDGP effectively and efficiently for proteins [25]. Previous approaches usually become numerically unstable as solution process progresses because the number of constraints to determine the coordinate of a new atom gets larger.

In this regard, we propose a heuristic algorithm, called the BetaMDGP, to maintain a constant number of constraints to determine the coordinate of a new atom. The BetaMDGP algorithm to the MDGP uses the beta-complex, which is a derivative geometric construct from the Voronoi diagram of atoms and effectively provides the proximity information among atoms [26–28]. Using the beta-complex, the BetaMDGP reduces the number of distance constraints required for determining new atom coordinate and thus finds the solution very efficiently. While the previous approaches are numerically unstable, the BetaMDGP provides the more stable solution compared to the previous algorithms because the BetaMDGP keeps the number of constraints constantly during the solution process. The BetaMDGP consists of two parts. First, we determine the coordinates for the centers of four nearby atoms to define the tetrahedral seed structure to start the process. Second, the BetaMDGP adds other atoms around the boundary of this determined substructure (at the beginning, it is the seed structure) using the beta-complex. The molecular structure is determined by sufficiently repeating this second procedure. It turns out that the proposed algorithm, in its current form, determines the protein structures very effectively and efficiently compared to existing algorithms. All figures of the molecular structures are created by the BetaMol program developed by the VDRC (http://voronoi.hanyang.ac.kr) that is free to download [29].

2 Methods

The proposed algorithm is based on three geometric constructs: the Voronoi diagram of atoms, the quasi-triangulation, and the beta-complex. Consider the set $P = \{p_1, p_2, \ldots, p_n\}$ where $p_i \in P$ is a point in three-dimensional space. The ordinary Voronoi diagram VD of P is the tessellation of the space with a set of n Voronoi cells (V-cells) where the V-cell VC(p_i) is the set of the locations that are closer to p_i than to the others. Consider the set $A = \{a_1, a_2, \ldots, a_n\}$ where $a_i = (p_i, r_i) \in A$ is a spherical atom with the center p_i and radius r_i in three-dimensional space. The Voronoi diagram $\mathcal{VD}$ of A is the tessellation of the space with a set of n V-cells where the VC(a_i) is the set of the locations that are closer to the boundary of a_i than to the boundary of any other atom. $\mathcal{VD}$ is more formally called the additively weighted Voronoi diagram in computational geometry and is different from the ordinary Voronoi diagram of points VD. $\mathcal{VD}$ can be represented as $\mathcal{VD} = (V^{\mathcal{V}}, E^{\mathcal{V}}, F^{\mathcal{V}}, C^{\mathcal{V}})$ where the Voronoi vertex (V-vertex) $v^{\mathcal{V}} \in V^{\mathcal{V}}$ corresponds to the center of the empty sphere tangent to the boundaries of four nearby atoms; the Voronoi edge (V-edge) $e^{\mathcal{V}} \in E^{\mathcal{V}}$ corresponds to the locus of the center of the empty sphere tangent to the boundaries of three nearby atoms; the Voronoi face (V-face) $f^{\mathcal{V}} \in F^{\mathcal{V}}$ corresponds to the locus of the center of the empty sphere tangent to the boundaries of two nearby atoms; the V-cell $c^{\mathcal{V}} \in C^{\mathcal{V}}$ corresponds to an atom. The topology among the V-vertices, V-edges, V-faces, and V-cells in $\mathcal{VD}$ are usually maintained in a radial-edge data structure [30]. $\mathcal{VD}$ can be computed in $O(n^3)$ time for general spherical balls in the worst case but takes $O(n)$ time for molecular atoms on average. See [31] for $\mathcal{VD}$ and see [32] for the Voronoi diagram in general.

Applications of the Voronoi diagram use the traversal on its topology structure, and the dual of the Voronoi diagram is frequently used for this purpose because it simplifies the traversal algorithms [33, 34]. The dual structure of the ordinary Voronoi diagram VD is well-known as the Delaunay triangulation which has many powerful properties primarily for it being a simplicial complex [32]. However, the dual of the Voronoi diagram of atoms $\mathcal{VD}$, now known as the quasi-triangulation $\mathcal{QT}$, was recently defined and characterized by Kim and colleagues as follows. $\mathcal{QT} = (V^{\mathcal{Q}}, E^{\mathcal{Q}}, F^{\mathcal{Q}}, C^{\mathcal{Q}})$ where $v^{\mathcal{Q}} \in V^{\mathcal{Q}}$ is mapped from $c^{\mathcal{V}} \in C^{\mathcal{V}}$; $e^{\mathcal{Q}} \in E^{\mathcal{Q}}$ is mapped from $f^{\mathcal{V}} \in F^{\mathcal{V}}$; $f^{\mathcal{Q}} \in F^{\mathcal{Q}}$ is mapped from $e^{\mathcal{V}} \in E^{\mathcal{V}}$; $c^{\mathcal{Q}} \in C^{\mathcal{Q}}$ is mapped from $v^{\mathcal{V}} \in V^{\mathcal{V}}$. Note that all the simplexes in $\mathcal{QT}$ are mapped from the simplexes in $\mathcal{VD}$ and all the mappings are one-to-one. The conversion between $\mathcal{VD}$ and $\mathcal{QT}$ can be done in $O(m)$ time in the worst case where m represents the number of simplexes in $\mathcal{QT}$. $\mathcal{QT}$ is known to have a phenomenon called an anomaly. For the details of $\mathcal{QT}$, see [35, 36, 27, 37].

The beta-complex corresponding to the real-value β is a subset of $\mathcal{QT}$ such that every simplex σ in $\mathcal{QT}$ is removed if a spherical probe of radius β can pass through σ without intersecting the atoms corresponding to it. Hence, each simplex in the beta-complex represents the proximity among some atoms within the molecular boundary. The beta-shape is defined by the region of the space bounded by the boundary of the beta-complex. Hence, the boundary of the beta-shape determines the proximity among the atoms on the molecular boundary with respect to the probe. We emphasize here that the beta-complex can be computed very efficiently from the quasi-triangulation, and its correctness is mathematically guaranteed. For the details, see [28, 26, 27].

Fig. 1 illustrates the idea of these geometric constructs in the plane. Fig. 1(a) shows a two-dimensional molecule A consisting of nine atoms. Fig. 1(b) is the Connolly surface of A corresponding to the black circular probe. Note that there

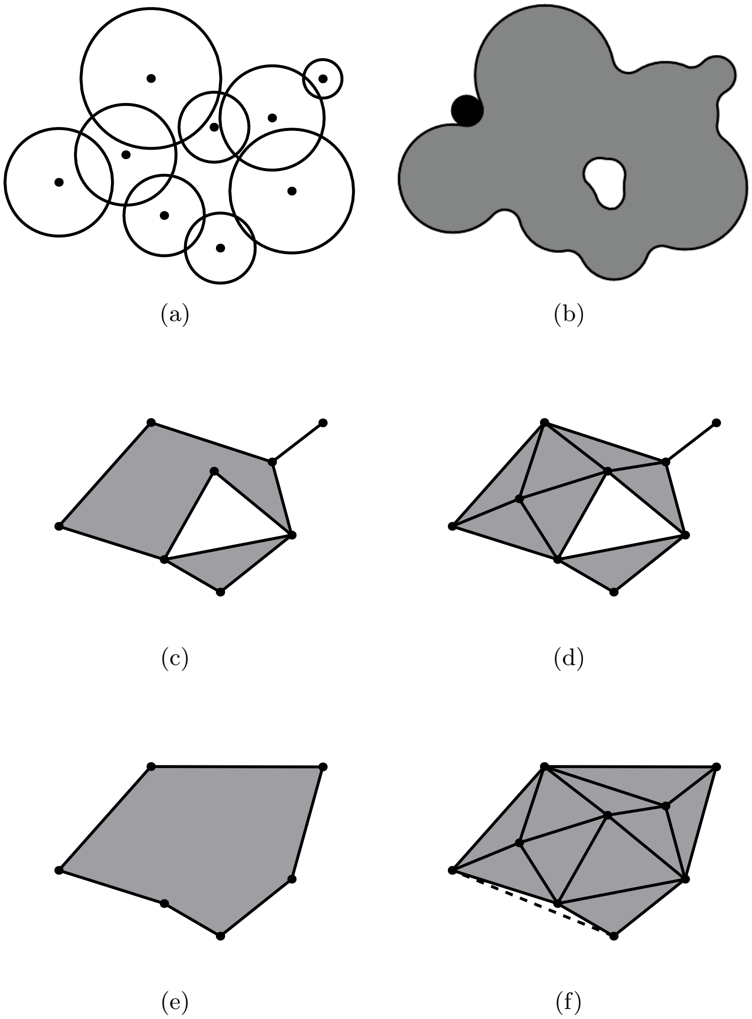


Fig. 1. Molecule, the Connolly surface, the beta-shape, and the beta-complexes in 2D. (a) A molecule (9 atoms), (b) the Connolly surface corresponding to a small probe, (c) the corresponding beta-shape, (d) the corresponding beta-complex, (e) a beta-shape corresponding to a larger probe, and (f) the corresponding beta-complex.

is an interior void. Fig. 1(c) shows the beta-shape corresponding to the Connolly surface of Fig. 1(b). The beta-shape has an interior void corresponding to the void of the Connolly surface and a dangling edge corresponding to a pair of atoms that are exposed to or touched by the probe. The boundary of the beta-shape has 8 vertices and 10 edges (7 on the exterior boundary and 3 on the interior void). Fig. 1(d) shows the corresponding beta-complex. Note that each vertex of the beta-shape and beta-complex corresponds to an atom. Fig. 1(e) and (f) show the beta-shape and the beta-complex corresponding to a larger probe, respectively. Note that both the dangling edge and the internal void have now disappeared. The dotted line segments in Fig. 1(f) together with the simplexes of the beta-complex form the quasi-triangulation of the molecule.

Based on these three constructs, the proposed BetaMDGP algorithm grows a molecular structure by adding one atom at a time that is selected by using the beta-complex for an appropriate value of the probe radius β. In the proposed algorithm, we start from tetrahedron τ consisting of four atoms which are guaranteed to be in a close neighborhood in a certain sense that will be described below. Then, we grow the structure by adding one shell of nearby atoms. Thus, we call the idea of this algorithm *"shell-growing."*

We first consider a two-dimensional example shown in Fig. 2. Suppose that Fig. 2(a) shows a true two-dimensional molecular structure that is stored in an NMR file. We first choose three nearby atoms which must form a (red-colored) seed triangle $t_0 = (a_1, a_4, a_{10})$ consisting of the centers of the three atoms a_1, a_4, and a_{10} in Fig. 2(b). The triangle t_0 can be determined by arbitrarily choosing one atom, say a_1, and two nearby atoms by looking at the distances to a_1. Let $T_0 = \{t_0\}$ and compute $\mathcal{BC}(T_0)$ whose boundary $\partial\mathcal{BC}(T_0)$ has three edges (the red chain in Fig. 2(b)). In this particular case, $\partial\mathcal{BC}(T_0)$ coincides with the boundary of the seed triangle t_0. We call $\partial\mathcal{BC}(T_0)$ the *shell* Sh_0 of T_0. Then, for each edge of Sh_0, we define another triangle by choosing another atom closest to two atoms consisting of the edge. After we determine the additional three triangles in such a fashion, say t_1, t_2, and t_3, we get Fig. 2(b). We call this operation shell-growing. Let $T_1 = \{t_0, t_1, t_2, t_3\}$. Then, we compute the beta-complex $\mathcal{BC}(T_1)$ for some value of β as shown in Fig. 2(c). Consider the red-colored $\partial\mathcal{BC}(T_1)$ the *shell* Sh_1 of T_1. Applying the shell-growing process once more by adding a new triangle for each edge $e \in Sh_1$, we get another set T_2 as shown in Fig. 2(d). Then, Fig. 2(e) shows the beta-complex $\mathcal{BC}(T_2)$ as well as T_2. Fig. 2(f) shows the last step of this model construction process to add the last atom a_7 which is under-determined. Note that a_7 can be placed at either a_7' or a_7'' without violating the distance constraint. Hence, there can be multiple solutions in the MDGP depending on the condition of the distance constraints in the input data. In such a case, however, adding another constraint on such an atom can uniquely determine the molecular structure. It is notable that such under-determined situations frequently arise in real NMR files.

Suppose that ∂T denotes the boundary of the union of the underlying space taken by each triangle $t \in T$. Then, $\partial\mathcal{BC}(T)$ may or may not be identical to ∂T. For example, Fig. 2(c) shows that ∂T_1 has six vertices but $\partial\mathcal{BC}(T_1)$ has five

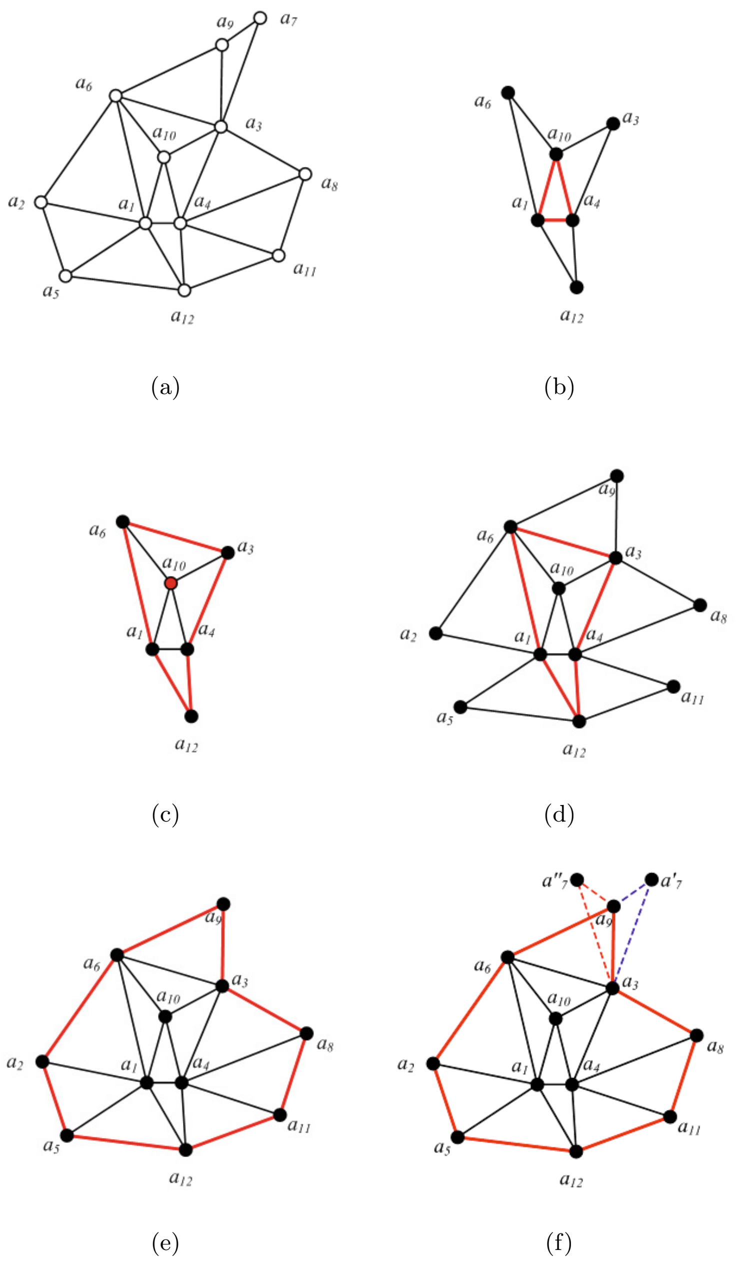


Fig. 2. The idea of the BetaMDGP algorithm in two-dimensional space. (a) The true structure to determine; (b) $T_0 = \{t_0\}$ (t_0 is the (red-colored) seed triangle) and $T_1 = \{t_0, t_1, t_2, t_3\}$; (c) $\mathcal{BC}(T_1)$ and (red-colored) $\partial\mathcal{BC}(T_1)$; (d) T_2; (e) $\mathcal{BC}(T_2)$; and (f) a multiple solution case in the MDGP.

vertices only: a_{10} does not appear on $\partial\mathcal{BC}(T_1)$. Therefore, when we compute T_2, we can safely ignore a_{10} from further consideration. In Fig. 2(e), if we compute $\partial\mathcal{BC}(T_2)$, we can now ignore three atoms from further consideration (i.e., a_1, a_4, and a_{10}). This reduction can contribute to the solution quality because it simplifies the solution process by removing the conflicting constraints as much as possible. It also contributes to algorithmic efficiency. The accumulation of the round-off error does not occur in the BetaMDGP, and thus the numerical stability is also improved. The three-dimensional MDGP can be similarly solved using the three-dimensional beta-complex. According to our experiment, the number of omitted atoms for the three-dimensional MDGP is significant.

Now, we consider a three-dimensional example of the BetaMDGP. See Fig. 3. Suppose that Fig. 3(a) shows the true three-dimensional molecular structure stored in an NMR file. We start the process with a seed tetrahedron $\tau_0 = (a_1, a_4, a_5, a_{11})$ consisting of the centers of four closely located atoms a_1, a_4, a_5, and a_{11}. The BetaMDGP algorithms grows τ_0 (the gray tetrahedron in Fig. 3(b)) by adding one shell of atoms around the current $T = \{\tau_0\}$ as follows: i) Compute the beta-complex of the current T for the appropriate β-value, ii) find the set ΔT of the new tetrahedron added to T for the faces on the boundary of the beta-complex $\partial\mathcal{BC}(T)$, and iii) $T = T \cup \Delta T$. Repeating this procedure a sufficient number of times correctly determines the structure of a molecule from the NMR data. In this paper, we use $\beta = 1.4$Å, which corresponds to the radius of the probe for a water molecule. The gray tetrahedron in Fig. 3(b) shows the seed tetrahedron τ_0. Let $T_0 = \{\tau_0\}$. The beta complex $\mathcal{BC}(T_0)$ is identical to τ_0. Then, for each face of τ_0, we define another tetrahedron by choosing the other atom closest to the vertices of the face. After we determine the additional four tetrahedron, say τ_1, τ_2, τ_3, and τ_4, we get Fig. 3(c) as the *shell-growing*. Let $T_1 = \{\tau_0, \tau_1, \tau_2, \tau_3, \tau_4\}$. Then, we compute the beta-complex $\mathcal{BC}(T_1)$ of T_1 for some value of β, as shown in Fig. 3(d). Consider $\partial\mathcal{BC}(T_1)$, i.e. the *shell* Sh_1 of T_1. Applying the shell-growing process once more using the faces on $\partial\mathcal{BC}(T_1)$, we get another set T_2 as the new tetrahedron added to T_1 for each face $f \in \partial\mathcal{BC}(T_1)$ as shown in Fig. 3(e). $\partial\mathcal{BC}(T_2)$ becomes Sh_2 as shown in Fig 3(f).

Suppose that ∂T denotes the boundary of the union of the underlying space taken by each tetrahedron $\tau_i \in T$. Then, like its two-dimensional counterpart, $\partial\mathcal{BC}(T)$ may or may not be identical to ∂T. For example, Fig. 3(d) shows that ∂T_1 has eight vertices but $\partial\mathcal{BC}(T_1)$ has seven vertices only: a_1 does not appear on $\partial\mathcal{BC}(T_1)$. Therefore, when we compute T_2, we can ignore a_1 from further consideration.

The following algorithm briefly describes the three-dimensional BetaMDGP algorithm. The input of the BetaMDGP algorithm is an atom set A where $a_i = (p_i, r_i) \in A$ is an atom with the unknown center p_i (but its radius r_i is known) and the distance set D where its element $d_{i,j} < \rho_{cutoff}$ is the inter-atomic distance between a_i and a_j. We used the usual cutoff distance 5 Å in order to simulate the NMR data. The output of the BetaMDGP algorithm is the atom set $\tilde{A}$ where $\tilde{a_i} = (\tilde{p_i}, r_i) \in \tilde{A}$ is an atom with the known coordinate of the center $\tilde{p_i}$. Step 1 determines the seed tetrahedron with the coordinates of the constituting

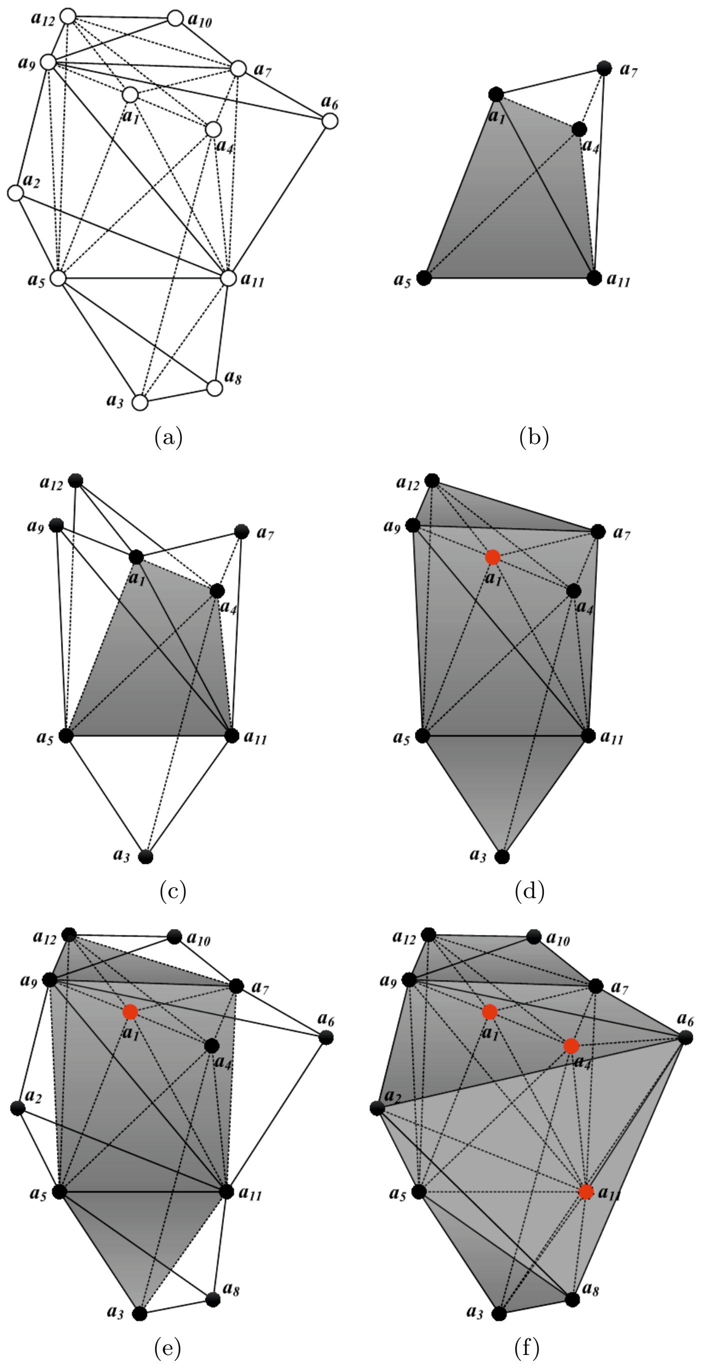


Fig. 3. The idea of the BetaMDGP algorithm in the three-dimensional space. (a) The true structure to determine; (b) $T_0 = \{\tau_0\}$ (τ_0 is the (gray-colored) seed tetrahedron) and τ_1 added to T_0; (c) $T_1 = \{\tau_0, \tau_1, \tau_2, \tau_3, \tau_4\}$; (d) $\mathcal{BC}(T_1)$; (e) T_2; and (f) $\mathcal{BC}(T_2)$

atoms. Step 2 performs the sell-growing procedure to determine as many atoms as possible. A newly determined atom a_p has three distances from the three atoms of f_p on $\partial\mathcal{BC}(\tilde{A})$. The average distance among a_p to the three atoms of f_p is $dist(a_p, f_p)$. However, the shell-growing procedure may not be able to exhaust all the atoms because there can be some atoms (which are called under-determined) where each does not have four distances from the atoms on $\partial\mathcal{BC}(\tilde{A})$. If such an under-determined atom exists, we choose an arbitrary location as long as it does not violate both its distance constraints and the well-packed molecular structure property.

```
Algorithm. Three-dimensional BetaMDGP
Input:
        $A = \{a_1, a_2, \ldots, a_n\}$ where $a_i = (p_i, r_i) \in A$ is an atom with the
              unknown center $p_i$ and the known radius $r_i$ of a particular type
              in the NMR file
        $D = \{d_{i,j} | d_{i,j}$ the distance between $a_i$ and $a_j$, $d_{i,j} < \rho_{cutoff}\}$
Output:
        $\tilde{A} = \{\tilde{a_1}, \tilde{a_2}, \ldots, \tilde{a_n}\}$ where $\tilde{a_i} = (\tilde{p_i}, r_i)$ with the known coordinate $\tilde{p_i}$
Step 1. Initialization:
    Step 1.1.  Make a seed tetrahedron $\tau_0$ with four nearby atoms in $A$.
    Step 1.2.  Insert the four atoms of $\tau_0$ to $\tilde{A}$.
    Step 1.3.  Determine the coordinates of the four atoms in $\tilde{A}$.
    Step 1.4.  $A \leftarrow A - \tilde{A}$
Step 2. Shell-growing: While $A \neq \emptyset$,
    Step 2.1.  Compute the beta-complex $\mathcal{BC}$ of $\tilde{A}$.
    Step 2.2.  Find the set $\mathcal{F}_\beta$ of the faces on $\partial\mathcal{BC}(\tilde{A})$.
    Step 2.3.  While $\mathcal{F}_\beta \neq \emptyset$,
            - Get a face $f_p \in \mathcal{F}_\beta$, $\mathcal{F}_\beta \leftarrow \mathcal{F}_\beta - \{f_p\}$.
            - Get an atom $a_p \in A$ which has three distances from the three
                  atoms of $f_p$, $dist(a_p, f_p)$ is the shortest from $f_p$.
            - $\tilde{A} \leftarrow \tilde{A} + \{a_p\}$ and determine the coordinate of $a_p$.
            - $A \leftarrow A - \{a_p\}$
            - If $A = \emptyset$, terminate the shell-growing process.
    End-while.
    Step 2.4.  If such $a_p$ does not exist,
            - Go to Step 3.
    End-if.
End-while.
Step 3. Marginal process: While $A \neq \emptyset$,
    - Get an atom $a_i \in A$.
    - $\tilde{A} \leftarrow \tilde{A} + \{a_i\}$ and determine the coordinate of $a_i$ by using the distance
          related $a_i$.
    - $A \leftarrow A - \{a_i\}$
End-while.
Step 4. Terminate.
```

3 Results

The proposed BetaMDGP algorithm has been validated through implementation and testing with data obtained from the PDB files. The input files contain the atomic pairs whose inter-atomic distances fall within the usual cutoff distance of 5Å. The experiment of the BetaMDGP algorithm shows extremely good solution quality in that the recovered structures are very close to the original PDB structures from geometric measures points of view such as the RMSD between the equivalent atoms in both the original PDB models and the reconstructed models, the existence and distribution of the interior voids, and the distribution of the covalent bond lengths. All the experimental results were visualized using the BetaMol program [29]. Note that all the reconstructed structures are displayed after it was superposed with the original PDB model using the structure superposition program, the BetaSuperpose [38]. We tested the BetaMDGP algorithm with three types of NMR data created from the PDB files: i) data without an interval (all atom types), ii) data with an interval (all atom types), and iii) data with hydrogen atoms with an interval. The computational environment is as follows: Intel Core2 Duo E6550 CPU and 4 GB memory on a Windows 7 Ultimate platform.

As the first test, we created NMR files according to the inter-atomic distances within a 5Å cutoff radius. In other words, we computed all the pairwise inter-atomic distances for all the atoms in each PDB file and output the atom pairs with an inter-atomic distance shorter than 5Å into an NMR file. See Fig. 4. The red structures in Fig. 4(a) and (d) show the true structures of 2lt8 (558 atoms) and 1xba (2068 atoms) in the PDB after we removed all the hetero atoms and water molecules. Note that 2lt8 in Fig. 4(a) was determined by NMR spectroscopy and therefore it is one (to be specific, the first one) of the ensembles. 1xba in Fig. 4(d) is from the X-ray crystallography. The blue structure in Fig. 4(b) shows the reconstructed structure by the BetaMDGP algorithm using the input file from the 2lt8. From the visual inspection, we can see that both Fig. 4(a) and (b) are very similar. Fig. 4(c) shows the ribbon models of both the structures after they were superposed. This figure shows that the backbones are almost identical. Fig. 4(e) and (f) are the reconstructed structure and the ribbon models for the 1xba model, respectively. The reconstruction for 1xba also has a similar shape as its original PDB model.

We also checked the interior structures by computing the voids. A void is a cavity in a molecular interior that is accessible to some molecule and is important for understanding the molecular characteristics. Fig. 5(a) shows the distribution of the interior voids of the PDB structure 1xba. The dark red color denotes the voids where a spherical probe with the radius 1.4Å (corresponding to a water molecule) can be placed. Fig. 5(b) shows the same information for the reconstructed structure. Note the similarity of the void distribution for the water molecules. Fig. 5(c) and (d) show the distribution of the voids corresponding to a probe with the radius 1.0Å. Both the original structures and the reconstructed structures are remarkably similar!

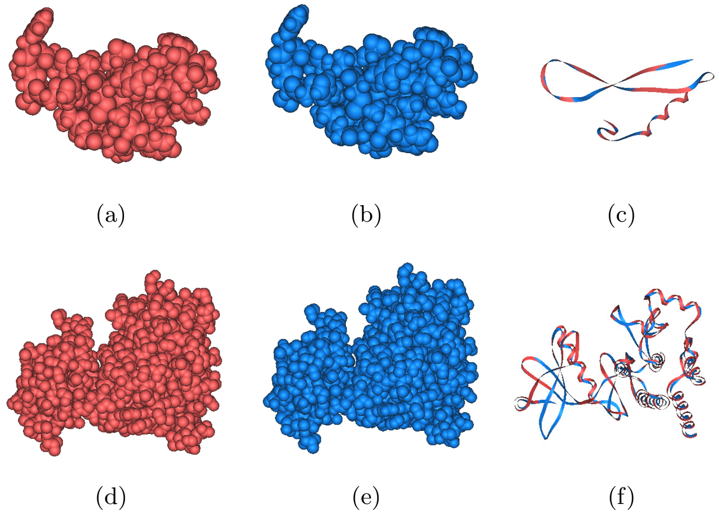


Fig. 4. Structure comparison. (a) and (d) the PDB structures 2lt8 (558 atoms) and 1xba (2068 atoms), respectively; (b) and (e) the reconstructed structures by the BetaMDGP; and (c) and (f) the ribbon models for the backbones of the true (red) and reconstructed (blue) structures after they were superposed.

For validation of the solution quality of the proposed algorithm, visual inspection is of course insufficient. We statistically checked the solution quality as well. First, we checked the root mean squared deviation (RMSD) between the PDB models and the reconstructed models as follows. Let $dist_i^{pdb2beta}$ be the distance between an atom a_i in PDB and its reconstructed atom by using the BetaMDGP algorithm. The RMSD for n atoms is given as

$$\text{RMSD} = \sqrt{\frac{1}{n}\sum_{1}^{n}(dist_i^{pdb2beta})^2}. \tag{6}$$

Table 1 shows the statistics of the RMSDs and the computation time. Column A is the PDB ID of the original PDB models used in the first test and the number of atoms is in column B. Columns C and D are the number of residues and ensembles, respectively. Note that the 1xba model determined from X-ray crystallography has no ensemble. Column E shows the statistics of the RMSDs between the original PDB models and its reconstructions. Note that the 2lt8 and 2jwu models were determined by NMR spectroscopy. Hence, we reported the average value of the RMSDs (E3) for each ensemble after the reconstructions of all the ensemble instances were separately computed by the BetaMDGP program. Similarly, we reported the minimum (E1), the maximum (E2), and the standard deviation (E4) value of the RMSDs. The average RMSDs (E3) for 2lt8, 2jwu,

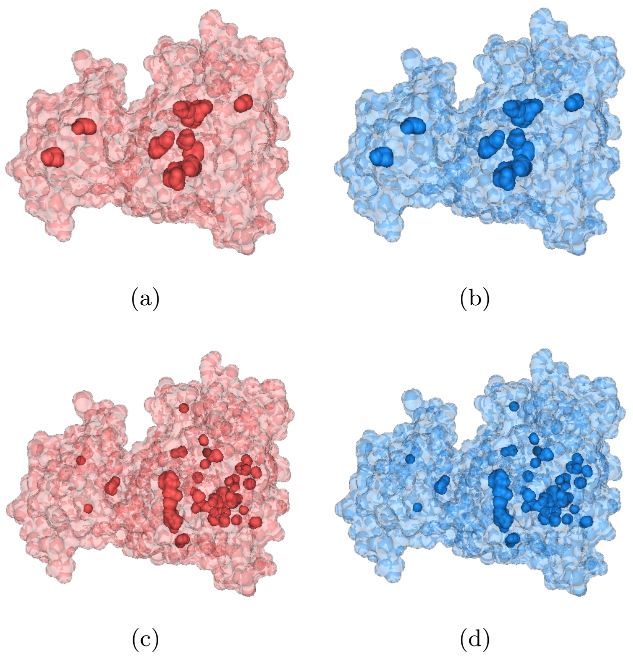


Fig. 5. Void distributions of 1xba. (a) and (c) the interior voids for the PDB structures and (b) and (d) the interior voids for the reconstructed structures ((a) and (b): $\beta = 1.4$, (c) and (d): $\beta = 1.0$).

Table 1. Statistics of the RMSDs and the computation times from the BetaMDGP

PDB ID	#atoms	#res.	#ensem.	RMSD (Å) (E)				time(sec)
				min.	max.	avg.	stdev.	
(A)	(B)	(C)	(D)	(E1)	(E2)	(E3)	(E4)	(F)
2lt8	558	43	20	0.008	0.104	0.030	0.025	9.51
2jwu	922	56	20	0.001	0.208	0.017	0.046	7.62
1xba	2068	334	·	·	·	0.041	·	104.68

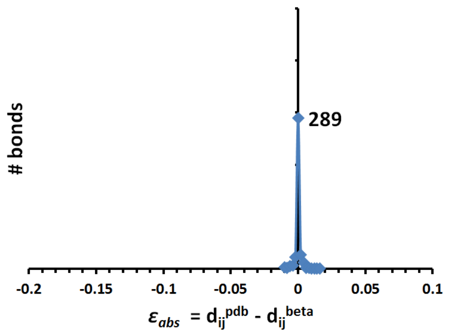
and 1xba were 0.030, 0.017, and 0.041Å, respectively. They are all tiny. The computation took 9.51, 7.62, and 104.68 sec, respectively (F). It currently seems relatively high because our current implementation of the Voronoi diagram and beta-complex algorithms are not optimally tuned for the MDGP problem. We expect this problem will be remedied in our future version with an expected computation reduction of tenfold or more.

We also checked the distributions of the covalent bond lengths in both the PDB and the reconstructed structures. Let d_{ij}^{pdb} and d_{ij}^{beta} be the length between the covalent bonded a_i and a_j in the PDB and the reconstructed model, respectively. Let $\epsilon_{abs} = d_{ij}^{pdb} - d_{ij}^{beta}$ be the absolute error. Fig. 6 shows the distribution of the absolute error ϵ_{abs} for the three examples. Note that all three graphs show that the distributions are extremely focused with a mean value of zero. We note that the volumes of the voids can also be computed and compared from a statistical point of view. From this test, we conclude that the proposed BetaMDGP algorithm reconstructs the original PDB structure effectively and efficiently.

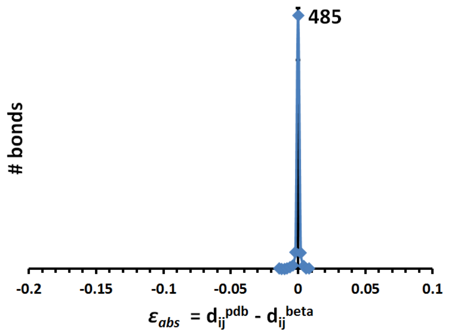
For the second test, we experimented with data consisting of the interval distances. We computed the inter-atomic distances from all the ensembles of the PDB model whose structures were determined from NMR spectroscopy. We created the lower and the upper bounds of the interval of each edge by the minimum value and the maximum value of the inter-atomic distances of the edges, respectively. Then, the medium value of interval shorter than 5Å was used as the input data for the test.

The red structures in Fig. 7(a) and (d) show the true structures of 2jmy (281 atoms, 19 models in the ensembles) and 2jwu (922 atoms, 20 models in the ensembles) in the PDB whose structures were determined from NMR spectroscopy. The red structures in Fig. 7 are one (to be specific, the model with the minimum RMSD after the superposition with the reconstruction) of the ensembles. We compared the reconstructed structure by the BetaMDGP with all the original ensemble structures. The blue structure in Fig. 7(b) shows the reconstructed structure by the BetaMDGP algorithm using the input files of the 2jmy. Fig. 7(c) shows the ribbon models for the superposed backbones of the original structure in Fig. 7(a) and the reconstructed structure in Fig. 7(b). Note that they are very close. Fig. 7(d), (e), and (f) are for the 2jwu model. Table 2 shows a summary of these experimental results. Column D denotes the number of models in the ensembles in the original PDB models. Column E shows the minimum of RMSD (E1), the maximum of RMSD (E2), the average of RMSD (E3), and the standard deviation of RMSD (E4) between each of the ensembles of the original model and the reconstructed model using the interval distance. The minimum RMSD (E1) between the reconstructed structure and 19 ensembles of 2jmy is 2.21, and the average RMSD (E3) is 2.34 Å. These RMSDs are obviously larger than the RMSD for the experiment with the data without an interval. This may be because we used the medium value of the interval distance as the input to the BetaMDGP program. From the second test, we also conclude that the BetaMDGP algorithm reconstructs the PDB structures at a fairly sufficient level of accuracy and efficiency.

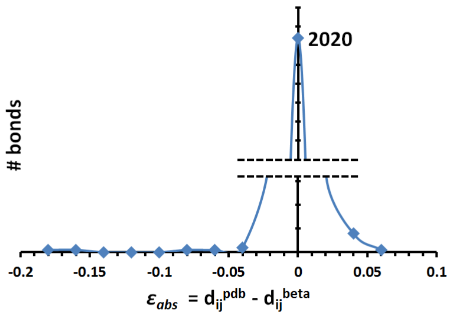
For the third test, we experimented the BetaMDGP algorithm for the input data consisting of only the hydrogen atoms with intervals for each distance-defined pair. We computed the inter-atomic distances between only the hydrogen atoms from all the ensembles of the PDB file. Then, we created the lower and the upper bounds of the interval by the minimum value and the maximum value of the inter-atomic distances, respectively. The medium value of interval shorter than 5Å is used as the input file.



(a)



(b)



(c)

Fig. 6. Difference in the covalent bond lengths between the original and the reconstructed structure. PDB ID: (a) 2lt8 (558 atoms), (b) 2jwu (922 atoms), and (c) 1xba (2068 atoms).

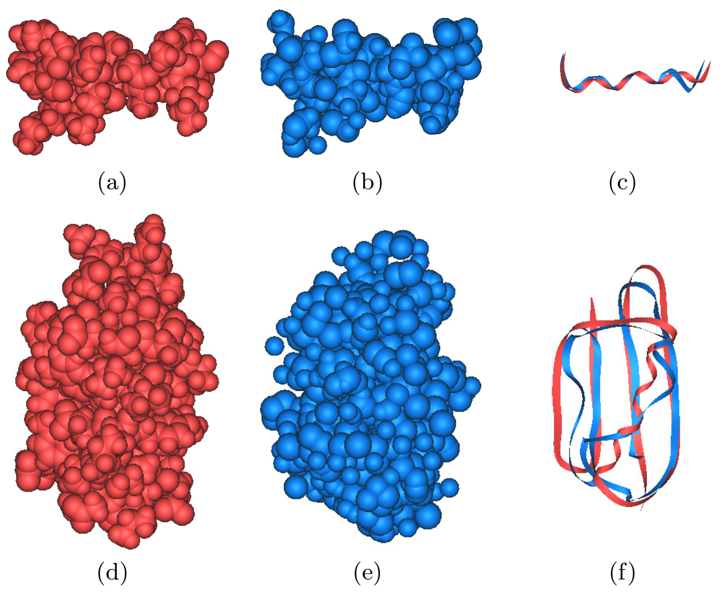


Fig. 7. Structure comparison. (a) and (d) the original protein structures 2jmy and 2jwu, respectively; (b) and (e) the reconstructed structures by the BetaMDGP; and (c) and (f) the ribbon models of the original (red) and reconstructed (blue) structures after they were superposed.

Table 2. Summary of the RMSDs and the computation times from BetaMDGP with intervals

PDB ID	#atoms	#res.	#ensem.	RMSD (Å) (E)				time(sec)
				min.	max.	avg.	stdev.	
(A)	(B)	(C)	(D)	(E1)	(E2)	(E3)	(E4)	(F)
2jmy	281	15	19	2.21	2.59	2.34	0.10	1.972
2jwu	922	56	20	4.36	4.55	4.48	0.04	11.812

Table 3 shows a summary of this experimental result. In this experiment, we used the input distance of the two types: i) the medium value of the interval distance (Row I) and ii) the random value of each interval distance (Row II). The data in Row II are the results of 500 experiments. Column B is the number of hydrogen atoms and column E shows the statistics of the RMSDs between all the ensembles of the original model and the reconstructed model. The minimum RMSD (E1) of Row II (by the random choice) is significantly smaller than the minimum RMSD of Row I (by the medium choice). This implies that we may get better reconstruction if the distribution of the distances for each atom pair can be used. In this experiment, we used the RMSD as the measure of quality for the reconstructed models. However, we believe that this may not be an appropriate measure for the reconstructed model from the test data with intervals produced from the PDB ensemble. The development of an appropriate measure is an issue for further study.

Table 3. Statistics of the RMSDs and computation times from BetaMDGP with intervals (only hydrogen atoms)

	PDB ID	#atoms	#res.	#ensem.	RMSD (Å) (E)				time(sec)
					min.	max.	avg.	stdev.	
	(A)	(B)	(C)	(D)	(E1)	(E2)	(E3)	(E4)	(F)
I	2jmy	153	15	19	7.06	7.50	7.28	0.12	1.45
	2jwu	467	56	20	8.43	8.58	8.50	0.04	3.46
II	2jmy	153	15	19	3.69	13.41	6.85	1.08	1.33
	2jwu	467	56	20	5.64	16.37	9.42	1.48	3.21

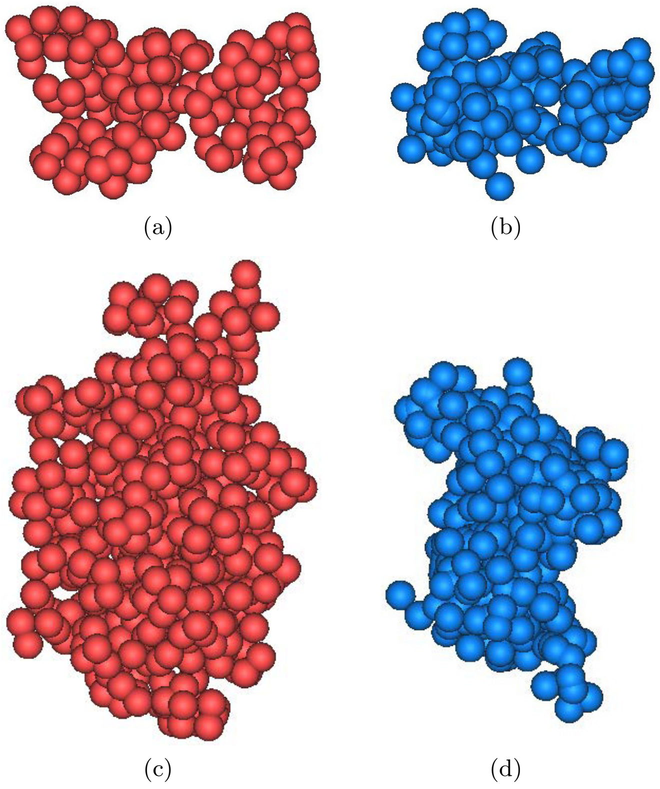


Fig. 8. Structure comparison. (a) and (c) the hydrogen atoms of the original protein structures of 2jmy and 2jwu; respectively and (b) and (d) the reconstructed structures by the BetaMDGP.

Fig. 8 shows the result of the experiments in Row II. The red model in Fig. 8(a) shows the true structures of 2jmy (153 hydrogen atoms) in the PDB. The blue model in Fig. 8(b) is the reconstructed model for the hydrogen atoms of 2jmy. The atoms in Fig. 8(b) are more closely positioned than the true structure in Fig. 8(a). Fig. 8(c) and (d) are for the 2jwu model (467 hydrogen atoms).

4 Discussions

The BetaMDGP algorithm was compared with other popular methods such as DGSOL and BP (Branch-and-prune) that we were able to benchmark. Other programs, for example, the geometric build-up algorithm were not available. First, the DGSOL is a program for solving MDGP based on the global continuation method with Gaussian smoothing of a merit function that only depends on the sparse distance data [9, 10] and can be freely downloaded from the DGSOL web site (http://www.mcs.anl.gov/more/dgsol/). In the current release, the DGSOL uses a variable-metric limited-memory code to trace the minimizers and can determine protein structures up to 200 atoms [10]. In this experiment, we also used the test data set obtained from the DGSOL site. Among the three available fragments consisting of 50, 100, and 200 atoms with a 1gpv structure (1840 atoms in total) from the PDB (The DGSOL provides only this one structure on the web site), we chose to test the biggest fragment with 200 atoms. The DGSOL determines the lower and the upper bounds of the distance intervals as follows. If $d_{i,j} = ||x_i - x_j||$ is the distance within the 5Å cutoff distance between atoms i and j, then the lower bound $l_{i,j} = d_{i,j}(1 - \varepsilon)$ and the upper bound $u_{i,j} = d_{i,j}(1 + \varepsilon)$ for some epsilon with $0 < \varepsilon < 1$. We ran both the BetaMDGP and the DGSOL using the various input data with intervals generated with different ε values.

Table 4 shows the test results of the BetaMDGP and the DGSOL. Columns A and B show the number of atoms and edges (i.e. the number of atom pairs with known distances in the input data), respectively. Column C shows the different ε values used for interval generation. Recall Eq. (5). Column D2 is what the DGSOL produced by Eq. (5) which describes how much the reconstructed structure satisfies the distance constraints with intervals. Column D1 shows the statistics using Eq. (5) from the reconstructed structure by the BetaMDGP. While the value of both the algorithms are tiny, the BetaMDGP values are smaller. Columns E1 and E2 show the RMSDs, defined by Eq. (6), from both BetaMDGP and DGSOL, respectively. Note that the RMSD of the BetaMDGP is significantly smaller than that of the DGSOL. The computation times in columns F1 and F2 show that the computation times from the BetaMDGP are significantly faster than the DGSOL.

Fig. 9 visually illustrates the experimental result of the case $\varepsilon = 0.16$ in Table 4. The red model in Fig. 9(a) is the segment of the original PDB model 1gpv that we extracted. It corresponds to the segment consisting of 200 atoms in the input file defined by the DGSOL website. Hence, this is the target structure that we want to reconstruct. The blue one in Fig. 9(b) and the green one in Fig. 9(c) are the reconstructions by the BetaMDGP and the DGSOL, respectively. Observe that the reconstruction by the BetaMDGP is very close to the original

Table 4. Comparison of the BetaMDGP and the DGSOL. The fragment used for the test has 200 atoms from the PDB structure 1gpv (1840 atoms in total). The parameter ε is used for the interval generation.

PDB ID: 1gpv			$p_{i,j}(x)$ (D)		RMSD (Å) (E)		time (sec) (F)	
#atoms (A)	#edges (B)	ε (C)	BetaMDGP (D1)	DGSOL (D2)	BetaMDGP (E1)	DGSOL (E2)	BetaMDGP (F1)	DGSOL (F2)
200	3300	0.04	0.001	0.007	0.004	5.395	1.273	22.256
200	3300	0.08	0.000	0.097	0.039	2.528	1.304	23.510
200	3300	0.12	0.000	0.000	0.085	5.055	1.394	22.097
200	3300	0.16	0.000	0.000	0.013	2.470	1.381	25.079

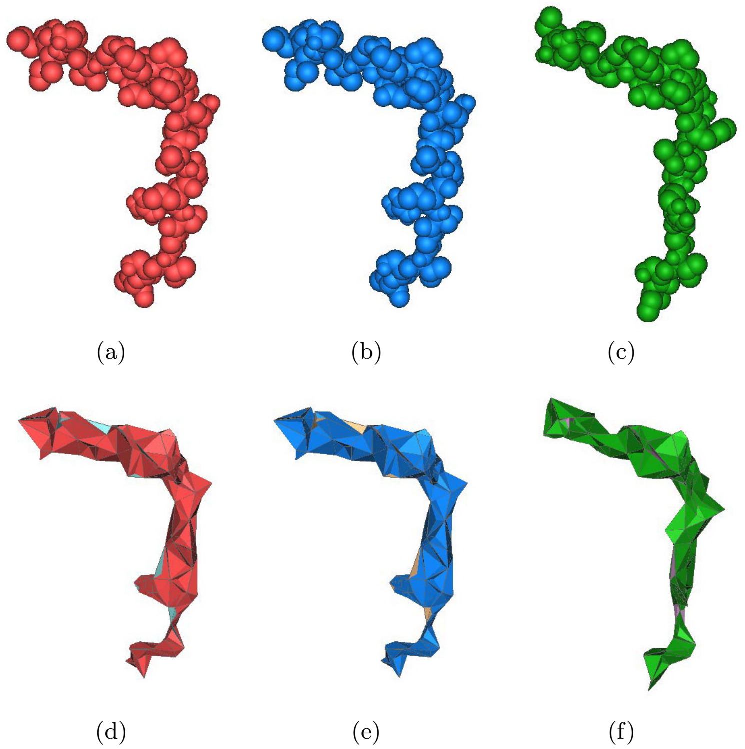


Fig. 9. Comparison of the structures reconstructed from the BetaMDGP and DGSOL against the original structure from PDB (1gpv). (a) the original protein structures (PDB code:fragment of 1gpv); (b) and (c) the reconstructed structures by the BetaMDGP and DGSOL, respectively; and (d), (e), and (f) the corresponding beta-shapes (β=1.4Å).

PDB model, whereas the one by the DGSOL is significantly different from the original model (The atom in the vertical column of the original PDB model does not exist). Fig. 9(d), (e), and (f) are the beta-shapes of the structures in Fig. 9(a), (b), and (c) where $\beta = 1.4$ Å respectively. The beta-shapes clearly show the reconstruction quality.

Second, the BetaMDGP was also compared with the MD-jeep program which implemented the Branch and Prune (BP) algorithm [25]. The MD-jeep (version 0.1) and test problems can be freely downloaded from http://www.antoniomucherino.it/en/mdjeep.php. The MD-jeep reconstructs the backbone structures through the formulation of a combinatorial optimization problem and uses a branch and prune strategy. To be compatible with the input data to the MD-jeep, we also generated an input file to the BetaMDGP from the same PDB files according to the rules used for the MD-jeep as follows. Given a PDB file, we first extracted N, C_α, and C atoms with their coordinates on a backbone. Then, the pairwise distances falling within 5 Å were computed in order to simulate NMR data as an input file to the BetaMDGP algorithm. We verified the identity of the MD-jeep input files downloaded from its web site and the generated BetaMDGP input files using the number of atoms and the interatomic distances. Running both the BetaMDGP and the MD-jeep algorithms produces their reconstructions which obviously contain only N, C_α, and C atoms, missing O and C_β (i.e., the first atom on each side-chain). Among the possible solutions found by MD-jeep, we used a solution that MD-jeep provided. We remark that the solution quality is likely to be improved if all MD-jeep solutions are used. Fig. 10(a), (b), and (c) visually show the ribbon models of the backbone of the original PDB structure (red), that of the BetaMDGP reconstruction (blue), and that of the MD-jeep reconstruction (green), respectively. Each reconstructed structure is displayed after it is superposed with the original one from the PDB file. Note that the BetaMDGP reconstructions are closer to the original models.

Given a backbone structure with known amino acid sequence information, it is possible to recover the entire protein structure by solving the side-chain prediction problem, abbreviated as the SCP-problem, which predicts the optimal conformation of the side-chains of all the amino acids in a protein. The general approach to the SCP-problem is to use a rotamer library which is derived by statistically clustering the observed side-chain conformations of known protein structures in the PDB [39–42]. The optimality is defined by the minimum potential energy of the protein structure determined by the conformation of all the side-chains where the energy is given by a forcefield. The SCP-problem is known as NP-hard [43–45] and is useful for flexible protein-ligand docking [46, 47] and homology modeling [48–50].

We generated the two types of missing atoms, O and C_β, with their coordinates. Then, we ran the BetaSCP program, also developed by the authors group [51], to get the entire protein structure of the backbones produced by both the BetaMDGP and the MD-jeep. Fig. 11(a) shows the structure of the original PDB files (red). Fig. 11(b) and (c) show the structures recovered by running the

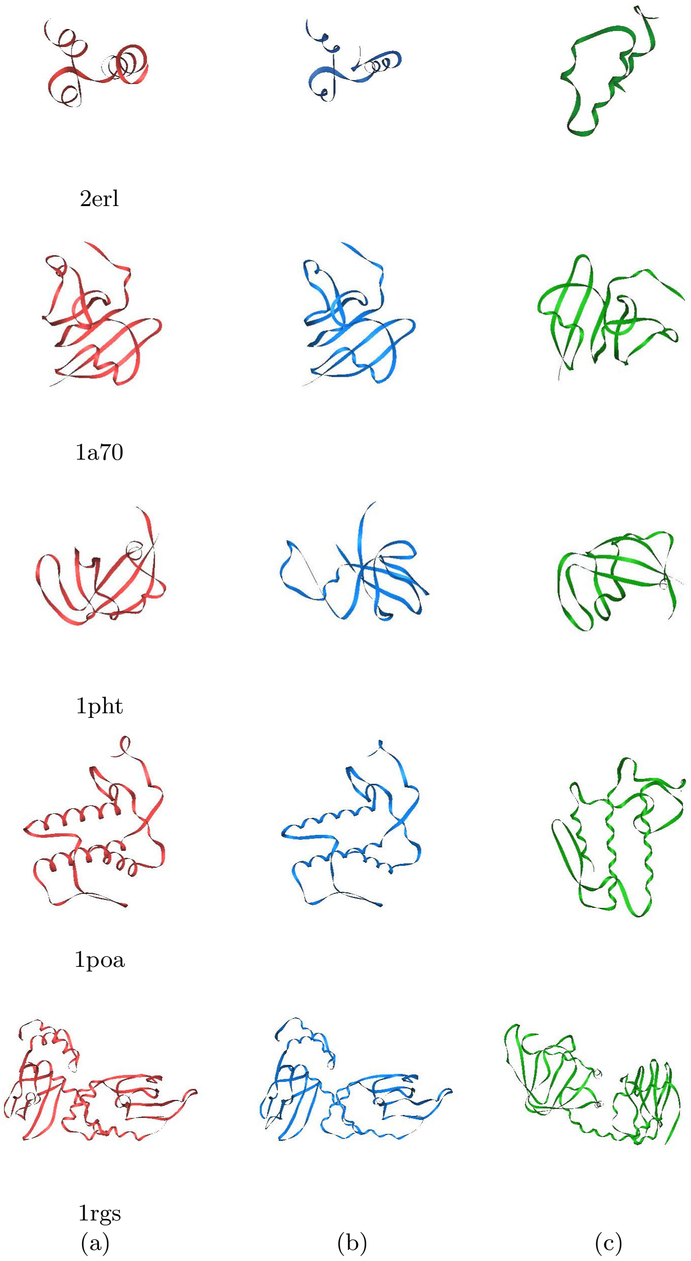


Fig. 10. Structure comparison with the ribbon models. (a) backbone of the original protein structures and (b) and (c) the reconstructed backbone structures by the BetaMDGP and the MD-jeep, respectively.

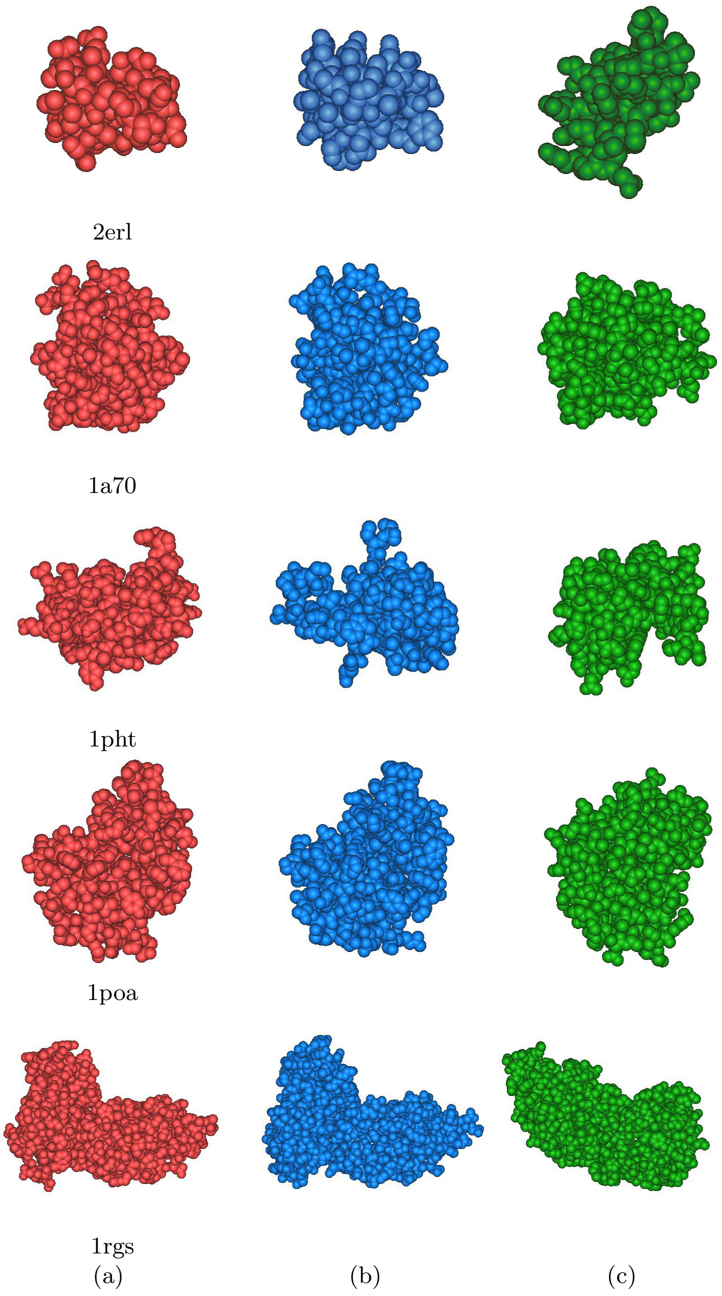


Fig. 11. Structure comparison. (a) the original protein structures and (b) and (c) the reconstructed structures by the BetaSCP program on the backbones from the BetaMDGP and the MD-jeep, respectively.

BetaSCP program on the backbones from the BetaMDGP (blue) and the MD-jeep (green), respectively. Observe that the result of the BetaMDGP is significantly better than that of the MD-jeep.

The visual result in Fig. 11 is statistically analyzed in Table 5. Row I corresponds to the BetaMDGP and Row II corresponds to the MD-jeep. Column B1 is the number of atoms of the original models and column B2 is the number of atoms in the backbones, both from the original PDB structures. Column D1 is the RMSD between the original backbone structure and the structure reconstructed by the BetaMDGP algorithm. Column D3 is the MD-jeep counterpart for column D1. Note that the backbone structures produced by the BetaMDGP are mostly better than those by the MD-jeep (with the exception 1pht). Column D2 is the RMSD between the entire original PDB structure and the reconstructed structure with the recovered side-chains. Column D4 is the MD-jeep counterpart for column D2. Note that the BetaMDGP solutions are mostly better than those of the MD-jeep. Columns E1 and E4 are the computation times for the reconstruction by the BetaMDGP and the MD-jeep, respectively. Columns E2 and E5 are the computation times for running the BetaSCP program. Columns E3 and E6 are the total computation times for solving the SCP problem after the BetaMDGP and the MD-jeep, respectively.

Table 5. Experimental statistics of the protein structures whose side-chains are recovered by the BetaSCP program on the backbones reconstructed by the BetaMDGP and MD-jeep (Row I: the BetaMDGP result; Row II: the MD-jeep result)

	PDB ID (A)	#atoms (B)		#residues (C)	RMSD (Å) (D)		time (sec) (E)		
		PDB (B1)	Backbone (B2)		BetaMDGP (D1)	Recon Entire Struct (D2)	BetaMDGP (E1)	BetaSCP (E2)	E1+E2 (E3)
I	2erl	566	120	40	0.75	0.88	4.81	1.22	6.03
	1a70	732	291	97	0.54	1.48	7.81	4.41	12.22
	1pht	810	249	83	2.16	2.95	6.14	5.16	11.30
	1poa	913	354	118	0.27	1.12	24.02	5.94	29.96
	1rgs	2015	792	264	0.67	1.98	35.46	21.34	56.80
					MD-jeep (D3)	Recon Entire Struct (D4)	MD-jeep (E4)	BetaSCP (E5)	E1+E2 (E6)
II	2erl	566	120	40	1.78	2.62	0.00	1.15	1.15
	1a70	732	291	97	2.10	3.01	0.01	4.29	4.30
	1pht	810	249	83	2.11	2.51	0.01	4.85	4.86
	1poa	913	354	118	2.22	2.83	0.01	5.64	5.65
	1rgs	2015	792	264	3.47	7.28	0.08	19.40	19.48

5 Conclusions

We proposed a new approach, the BetaMDGP, to the MDGP problem based on the beta-complex, which is a geometric construct derived from the Voronoi diagram of atoms. From experiments using simulated NMR files, the BetaMDGP reconstructs the original structures with surprising similarity except for the input data with interval distances. However, there are three main issues to be resolved in the future. First, the BetaMDGP algorithm needs to consider the interval distances as the current algorithm considers only the medium value of an interval. Second, we need to improve the BetaMDGP algorithm by considering the under-determined condition. The real NMR data may be more sparse than our test data. These NMR data cause a situation where the molecular structure cannot be determined by using only the input data. Therefore, we have to consider additional information to solve the under-determined condition. For example, we can consider additional information such as the chemistry information and produce the input distance from the under-determined atom using triangular inequality. Finally, we remark that the BetaMDGP algorithm needs improved computational efficiency and the convergence of the BetaMDGP algorithm with an optimization method such as the BP algorithm is likely to improve solution quality.

Acknowledgments. This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2011-0020410). C. Lavor and A. Mucherino were supported by the Brazilian research agencies FAPESP and CNPq. The authors thank Prof. G. Crippen for suggesting an important direction of this study with real NMR files and Dr. S.-W. Chi at KRIBB for the help with the NMR file format.

References

1. Donald, B.R.: Algorithms in Structural Molecular Biology. The MIT Press (2011)
2. Cavanagh, J., Fairbrother, W.J., Palmer III, A.G., Rance, M., Skelton, N.J.: Protein NMR spectroscopy: principles and practice. Academic Press (2006)
3. Jan, D.: Principals of protein X-ray crystallography. Springer (2006)
4. Blumenthal, L.M.: Theory and Applications of Distance Geometry. Oxford Clarendon Press (1953)
5. Crippen, G., Havel, T.: Distance Geometry and Molecular Conformation. John Wiley & Sons, New York (1988)
6. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. SIAM Review 56 (article in press, 2014)
7. Havel, T.F.: Distance Geometry, vol. 4. John Wiley & Sons (1995)
8. Saxe, J.: Embeddability of weighted graphs in k-space is strongly np-hard. In: Proceedings of 17th Allerton Conference in Communications Control and Computing, pp. 480–489 (1979)
9. Moré, J.J., Wu, Z.: Global continuation for distance geometry problems. SIAM Journal of Optimization 7, 814–836 (1997)
10. Moré, J.J., Wu, Z.: Distance geometry optimization for protein strucutures. Journal of Global Optimization 15(3), 219–234 (1999)
11. An, L.T.H.: Solving large scale molecular distance geometry problems by a smoothing technique via the gaussian transform and d.c. programming. Journal of Global Optimization 27, 375–397 (2003)
12. An, L.T.H., Tao, P.D.: Large-scale molecular optimization from distance matrices by a d.c. optimization approach. SIAM Journal of Optimization 14(1), 77–114 (2003)

13. Liberti, L., Lavor, C., Mucherino, A., Maculan, N.: Molecular distance geometry methods: from continuous to discrete. International Transactions in Operational Research 18, 33–51 (2010)
14. Wüthrich, K.: NMR in Structural Biology. World Scientific, New York (1995)
15. Havel, T.: An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. Progress in Biophysics and Molecular Biology 56(1), 43–78 (1991)
16. Hendrickson, B.: The Molecular Problem: Determining Conformation from Pairwise Distances. PhD thesis, Cornell University (1991)
17. Hendrickson, B.: The molecule problem: Exploiting structure in global optimization. SIAM Journal of Optimization 5, 835–857 (1995)
18. Dong, Q., Wu, Z.: A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. Journal of Global Optimization 26(3), 321–333 (2003)
19. Wu, D., Wu, Z.: An updated geometric build-up algorithm for solving the molecular distance geometry problems with sparse distance data. Journal of Global Optimization 37(4), 661–673 (2007)
20. Sit, A., Wu, Z., Yuan, Y.: A geometric buildup algorithm for the solution of the distance geometry problem using least-squares approximation. Bulletin of Mathematical Biology 71(8), 1914–1933 (2009)
21. Sit, A., Wu, Z.: Solving a generalized distance geometry problem for protein structure determination. Bulletin of Mathematical Biology, 1–28 (2011)
22. Brünger, A.T., Adams, P.D., Clore, G.M., DeLano, W.L., Gros, P., Grosse-Kunstleve, R., Jiang, J.S., Kuszewski, J., Nilges, M., Pannu, N.S., Read, R.J., Rice, L.M., Simonson, T., Warren, G.L.: Crystallography & nmr system: A new software suite for macromolecular structure determination. Acta Crystallographica Section D-Biological Crystallography D54, 905–921 (1998)
23. Schwieters, C.D., Kuszewski, J.J., Tjandra, N., Clore, G.M.: The xplor-nih nmr molecular structure determination package. Journal of Magnetic Resonance 160, 65–73 (2003)
24. Lavor, C., Liberti, L., Maculan, N., Mucherino, A.: The discretizable molecular distance geometry problem. Computational Optimization and Applications 52, 115–146 (2012)
25. Liberti, L., Lavor, C., Maculan, N.: A branch-and-prune algorithm for the molecular distance geometry problem. International Transactions in Operational Research 15, 1–17 (2008)
26. Kim, D.S., Cho, Y., Sugihara, K., Ryu, J., Kim, D.: Three-dimensional beta-shapes and beta-complexes via quasi-triangulation. Computer-Aided Design 42(10), 911–929 (2010)
27. Kim, D.S., Kim, J.K., Cho, Y., Kim, C.M.: Querying simplexes in quasi-triangulation. Computer-Aided Design 44(2), 85–98 (2012)
28. Kim, D.S., Seo, J., Kim, D., Ryu, J., Cho, C.H.: Three-dimensional beta shapes. Computer-Aided Design 38(11), 1179–1191 (2006)
29. Cho, Y., Kim, J.K., Ryu, J., Won, C.I., Kim, C.M., Kim, D., Kim, D.S.: BetaMol: a molecular modeling, analysis and visualization software based on the beta-complex and the quasi-triangulation. Journal of Advanced Mechanical Design, Systems, and Manufacturing 6(3), 389–403 (2012)
30. Cho, Y., Kim, D., Kim, D.S.: Topology representation for the Voronoi diagram of 3D spheres. International Journal of CAD/CAM 5(1), 59–68 (2005), http://www.ijcc.org
31. Kim, D.S., Cho, Y., Kim, D.: Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. Computer-Aided Design 37(13), 1412–1424 (2005)

32. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd edn. John Wiley & Sons, Chichester (1999)
33. Munkres, J.R.: Elements of Algebraic Topology. Perseus Press (1984)
34. Boissonnat, J.D., Yvinec, M.: Algorithmic Geometry. Cambridge University Press, Cambridge (1998)
35. Kim, D.S., Kim, D., Cho, Y., Sugihara, K.: Quasi-triangulation and interworld data structure in three dimensions. Computer-Aided Design 38(7), 808–819 (2006)
36. Kim, D.S., Cho, Y., Sugihara, K.: Quasi-worlds and quasi-operators on quasi-triangulations. Computer-Aided Design 42(10), 874–888 (2010)
37. Kim, D.S., Cho, Y., Ryu, J., Kim, J.K., Kim, D.: Anomalies in quasi-triangulations and beta-complexes of spherical atoms in molecules. Computer-Aided Design 45(1), 35–52 (2013)
38. Kim, J.K., Kim, D.S.: Betasuperposer: Superposition of protein surfaces using beta-shapes. Journal of Biomolecular Structure & Dynamics 30(6), 684–700 (2012)
39. Dunbrack Jr., R.L.: Rotamer libraries in the 21st century. Current Opinion in Structural Biology 12(4), 431–440 (2002)
40. Dunbrack Jr., R.L., Karplus, M.: Backbone-dependent rotamer library for proteins. Journal of Molecular Biology 230(2), 543–574 (1993)
41. Dunbrack Jr., R.L., Karplus, M.: Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. Journal of Molecular Biology 1(5), 334–340 (1994)
42. Kono, H.: Rotamer libraries for molecular modeling and design of proteins. In: Park, S.J., Cochran, J.R. (eds.) Protein Engineering and Design (2009)
43. Chazelle, B., Kingsford, C., Singh, M.: The inapproximability of side-chain positioning. Technical report, Princeton University (2004)
44. Fung, H., Rao, S., Floudas, C., Prokopyev, O., Pardalos, P., Rendl, F.: Computational comparison studies of quadratic assignment like formulations for the In silico sequence selection problem in De Novo protein design. Journal of Combinatorial Optimization 10(1), 41–60 (2005)
45. Pierce, N.A., Winfree, E.: Protein design is NP-hard. Protein Engineering 15(10), 779–782 (2002)
46. Althaus, E., Kohlbacher, O., Lenhof, H.P., Müller, P.: A combinatorial approach to protein docking with flexible side-chains. In: RECOMB 2000 Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, pp. 15–24 (2000)
47. Althaus, E., Kohlbacher, O., Lenhof, H.P., Müller, P.: A combinatorial approach to protein docking with flexible side chains. Journal of Computational Biology 9(4), 597–612 (2002)
48. Lee, C., Subbiah, S.: Prediction of protein side-chain conformation by packing optimization. Journal of Molecular Biology 217(2), 373–388 (1991)
49. Tuffery, P., Etchebest, C., Hazout, S., Lavery, R.: A new approach to the rapid determination of protein side chain conformations. Journal of Biomolecular Structure & Dynamics 8(6), 1267–1289 (1991)
50. Leach, A.R.: Molecular Modelling: Principles and Applications. Prentice Hall (2001)
51. Ryu, J., Kim, D.S.: Protein structure optimization by side-chain positioning via beta-complex. Journal of Global Optimization (2012), doi: 10.1007/s10898-012-9886-3

Fast Calculation of the Empty Volume in Molecular Systems by the Use of Voronoi-Delaunay Subsimplexes

V.P. Voloshin[1], N.N. Medvedev[1,2], and A. Geiger[3]

[1] Institute of Chemical Kinetics and Combustion, SB RAS, 630090 Novosibirsk, Russia
nikmed@kinetics.nsc.ru
[2] Novosibirsk State University, Novosibirsk, Russia
[3] Physikalische Chemie, Technische Universität Dortmund, 44221 Dortmund, Germany
alfons.geiger@udo.edu

Abstract. The calculation of the occupied and empty volume in an ensemble of overlapping spheres is not a simple task in general. There are different analytical and numerical methods, which have been developed for the treatment of specific problems, for example the calculation of local intermolecular voids or − vice versa − of the volume of overlapping atoms. A very efficient approach to solve these problems is based on the Voronoi-Delaunay subsimplexes, which are special triangular pyramids defined at the intersection of a Voronoi polyhedron and Delaunay simplex. The subsimplexes were proposed in a paper [1] (*Sastry S.et al., Phys. Rev. E, vol.56, 5524–5532, 1997*) for the calculation of the cavity volume in simple liquids. Later, the subsimplexes were applied for the treatment of the union of strongly overlapping spheres [2] (*Voloshin V.P. et al., Proc. of the 8th ISVD, 170–176, 2011*). In this article we discuss wider applications of subsimplexes for the calculation of the occupied and empty volumes of different structural units, selected in molecular systems. In particular, we apply them to Voronoi and Delaunay shells, defined around a solute, as well as their intersection. It opens a way to calculate the components of the partial molar volume of a macromolecule in solution, what is important for the interpretation of experimental volumetric data for protein solutions. The method is illustrated by the application to molecular dynamics models of a hIAPP polypeptide molecule in water at different temperatures.

Keywords: Molecular dynamics simulation, solutions, bio-molecules, partial molar volume, Voronoi diagram, Delaunay simplex, molecular volume, occupied volume, empty volume.

1 Introduction

Ensembles of overlapping spheres are used as models for many real systems. In chemistry and biology the atoms are represented as van der Waals spheres, which are overlapping because of relatively short chemical bonds between them. In materials sciences packings of conglomerate particles are occurring in sandstones and colloids.

The problem of calculating the volume of a molecule was stated many years ago [3,4]. The local packing fraction is studied in the investigation of metal glasses [5]. A complementary task is to find the volume of intermolecular voids, which is important both for understanding protein structures and membrane permeability [6-8]. The volume of interatomic cavities is of interest in the theory of liquids [1] and for porous materials [9,10]. In recent years, there is a growing interest in the calculation of the volumetric characteristics of solutions. The density of water in the hydration shell, the density of the solute molecule, and the occurrence of additional voids in the boundary layer affect the partial molar volume (apparent volume) of a solute molecule in solution. The calculation of these components helps to interpret the data of volumetric measurements [11-15].

Many different methods are known to calculate the occupied and empty volumes in a system of spherical particles. Some of them use analytically derived formulas for the calculation of multiply intersecting spheres explicitly, or use numerical algorithms like Monte Carlo methods, see the papers cited in Refs. [2] and [16].

A novel approach was proposed in Ref. [1] and was applied to the calculation of the volume of interatomic cavities in a monatomic liquid. It substantially uses the Voronoi-Delaunay tessellation of the system. In this case a void between the atoms is composed of the empty volume of Delaunay simplexes. It was proposed to divide a simplex into smaller elements – orthogonal triangular pyramids (subsimplexes). The subsimplex has very important property: its occupied volume is defined by the only atom at the apex of the subsimplex. Thus, for any system of overlapping atoms, explicit formulas for the occupied volume of the subsimplex can be written. Then it was remained to sum (using a “rule of signs") the empty volumes of the subsimplexes which constitute the Delaunay simplexes of the cavity. (Note that the empty volume is obviously the difference between the total and the occupied volume). The authors of Ref. [1] used classical Voronoi-Delaunay tessellation, because atoms in their system have the same radius. However, they also emphasized that this approach can be implemented for spheres of various radii. In particular, the power (radical) decomposition can be used; but instead of the classical Delaunay simplexes, in this case the dual simplexes of the power Voronoi tessellation should be used.

In Ref. [2] the method of subsimplexes was implemented for the calculation of the volume of a union of overlapping spheres. This problem can be reduced to the determination of the occupied volume of the power Voronoi polyhedra in a system of overlapping spheres of different radii. Summing up the occupied volume of all subsimplexes, associated with a given atom, we find the desired occupied volume of the Voronoi polyhedron of this atom. (Here and below we use the terms Voronoi polyhedron (VP) and Delaunay simplex (DS) for both classical and power tessellations). In Ref. [2] we compared the method of subsimplexes with the other analytical method, which are known for the calculation of volume of a union of overlapping spheres. It was shown it is robust and even a bit faster then “a certified algorithm” [16].

In this paper, we propose to use subsimplexes for the calculation of occupied and empty volumes of various constructions of VP and DS. These may be the Voronoi region of a solute molecule in solution [17], or the Voronoi or Delaunay shells, given by the decomposition of a solution with respect to the solute molecule [15, 18].

Moreover, subsimplexes can help to estimate the intersection of Voronoi and Delaunay shells [19]. It helps to calculate the components of the partial molar volume of a macromolecule in solution, what is important for the interpretation of experimental volumetric data for protein solutions. We do not know other analytical approaches for such kind of constructions. Numerical methods (like Monte Carlo) are very slow for the studied problems. Our method is illustrated by the application to molecular dynamics models of a hIAPP polypeptide molecule in water at different temperatures.

2 Voronoi-Delaunay Subsimplex

2.1 Two Dimensions

Consider an atom A with its Voronoi polygon (VP) and the Delaunay simplex (DS) which corresponds to a vertex V of this polygon, Fig. 1. Join the points A and V by a line segment and draw perpendiculars from the point A to those polygon edges, which meet at the vertex V. For 2D there are only two such polygon edges. Let us denote the base points of the perpendiculars at these edges as E1 and E2. (Note, these perpendiculars coincide with DS edges.) The triangles AVE_1 and AVE_2 are the *Voronoi-Delaunay subsimplexes* (or simple *subsimplexes*) which correspond to the pair A - V. We will call them the duo of subsimplexes related to the pair A - V.

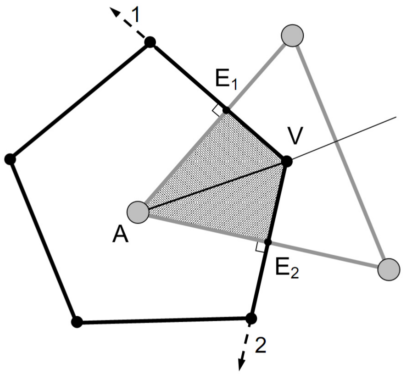
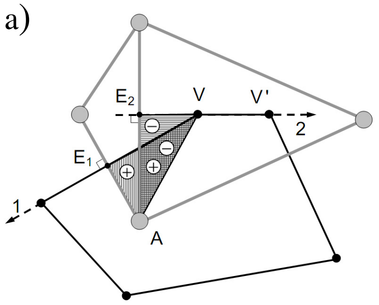


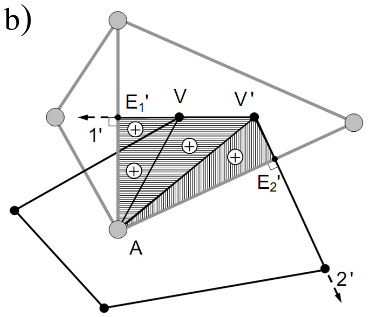
Fig. 1. 2D Voronoi-Delaunay subsimplexes for the Voronoi polyhedron of an atom A and the Delaunay simplex of its vertex V. Points E_1 and E_2 are the bases of the perpendiculars from the point A to the edges which meet at vertex V.

The example shown in Fig. 1 is typical for a more or less homogeneous system. In this case, the duo of subsimplexs represents the intersection of a VP and a DS. But such an ordinary configuration does not exhaust all situations in physical models. Fig. 2 shows an example, where one of the points E lies outside the VP. In this case the subsimplex AVE_2 of the pair A-V lies outside the Delaunay simplex of the vertex V (Fig. 2a). A piece of this subsimplex is also outside of the Voronoi polyhedron of the atom A.

Such a duo does not represent the intersection of the VP and the DS. However, if we consider two adjacent Delaunay simplexes together (of vertices V and V', Fig. 2b) we can find the correct volume of intersection of both simplexes and the polygon. In this case (Fig. 2c), by changing the sign of the exterior (inverted) subsimplex AVE_2, it can be compensated by the subsimplex $AV'E_1'$ (note, the points E_2 and E_1' are identical).

In Ref. [1] it was proposed to mark the volume of subsimplexes with different signs. In this example a sign factor S_E is defined, which is determined by the position of point E and the corresponding polygon edge relative to the vertex V. Point E_2 in Fig.2a is on the negative side of an axis (arrow 2), emanating from point V along the edge VV'. In other words, it lies on the other side of the point V than the edge of the polygon). In this case, for the subsimplex AVE_2 the factor is negative, $S_E = -1$. Else, if the point E lies on the edge (as in Fig. 1) or on its continuation, but on the positive side of the axis, emanating from the vertex, as in the case of point E'_1 and vertex V' in Fig. 2b) the factor is positive, $S_E = +1$.





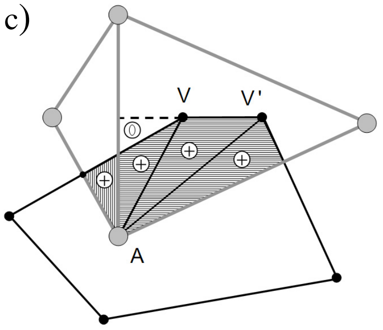


Fig. 2. Voronoi-Delaunay subsimplexes in the case of a non-ordinary configuration. (a) Vertex V, arrows 1 and 2 indicate the polygon edges, which are emanating from V. The point E_2 lies on the negative side of arrow 2 (opposite to the polygon edge emanating from vertex V). In this case the exterior subsimplex AVE_2 is considered negative, while the interior subsimplex AVE_1 is positiv. (b) Vertex V', arrows 1' and 2'. The point E_1' is outside the Voronoi edges, but in positive direction of the arrow (on the same side as edge 1' with respect to vertex V'). In this case, the subsimplex AV'E_1' is considered positive. (c) The sum of the (signed) subsimplexes at the vertices V and V' gives the intersection of these Delaunay simplexes and the Voronoi polygon.

The symbols "plus" and "minus" in Fig. 2a, b show the signs related with the considered subsimplexes. After summing, we will have an area corresponding to the intersection of the simplexes V and V' with the polygon A (Fig. 2c).

Recall that when dealing with power Voronoi decomposition, there may be cases when the center of atom A is outside its VP. This occurs when the position of atom A lies deep within the sphere characterizing another atom [20,21], see Fig. 2 in Ref. [2]. In this case point A lies on the other side of a VP face with respect to the VP itself. For all subsimplexes, which are based on this face, an additional factor $S_A = -1$ is defined. In all other cases, $S_A = +1$, see Fig. 6 in Ref. [2]. The final sign of the subsimplex in 2D is determined by the product of factors S_E and S_A .

2.2 Three Dimensions

Consider the VP of an atom A and the DS of its vertex V, Fig. 3. Three edges of the VP are incident to the vertex (let us denote them as 1, 2, 3), and there are three faces of the VP, which intersect at these edges (let us call them as (1,2), (1,3) and (2,3)). Connect the points A and V by a line segment, and draw perpendiculars from point A to these VP planes (as in 2D, the perpendiculars coincide with DS-edges). The points of intersection of the perpendiculars with the planes are denoted B_{12}, B_{13}, B_{23}.

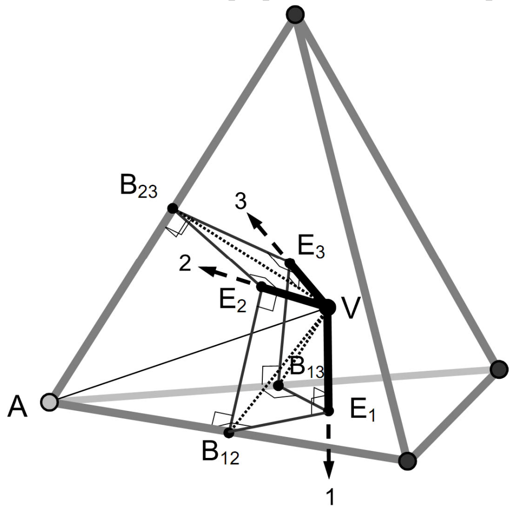


Fig. 3. Three-dimensional Voronoi-Delaunay subsimplexes. Point V is a vertex of the VP of atom A. Points B_{12}, B_{13} and B_{23} are the bases of the perpendiculars from A to the Voronoi faces, which meet in V. Points E_1, E_2 and E_3 are the bases of the perpendiculars from the points B_{12}, B_{13}, B_{23} to the VP edges 1, 2 and 3. The triangles (BVE) are the bases of subsimplexes with the vertex A. Six subsimplexes (sextet) are associated with each pair A - V.

In each of these planes we have a situation which is similar to the one discussed for 2D, where the point B (which is B_{12}, B_{13} or B_{23}) now plays the role of point A in Figs. 1, 2. Draw perpendiculars from the points B to the edges of their planes, Fig.3. Thes edges are the VP edges 1, 2, 3. The points of intersection of the perpendiculars with the polyherdon edges (or their continuations) are denoted also as E_1, E_2, E_3. Thus, on each face of the VP we got two right triangles. For example on the plane (1,2) these are $B_{12}VE_1$ and $B_{12}VE_2$. We will consider these triangles as the bases of pyramids with the vertex A as top. These pyramids form the Voronoi- Delaunay subsimplexes in 3D.

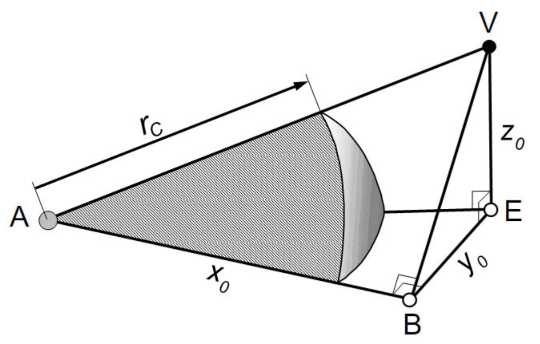


Fig. 4. 3D Voronoi-Delaunay subsimplex of the Voronoi polyhedron of an atom A and the Delaunay simplex of a vertex V of this polyhedron. Point B is the base of the perpendicular from vertex A to the face of the polyhedron. Point E is the base of the perpendicular from the point B to the VP edge, starting from vertex V. The region covered by atom A defines the occupied volume of the subsimplex.

The subsimplexes have a simple shape (Fig. 4), which made it possible to write an explicit expression for the calculation of the occupied volume inside subsimplexes. As discussed in Refs. [1, 2], this volume is completely determined by the atom centered on the vertex A. Some other atom can also overlap with the subsimplex (partially or fully), but this overlapping volume is always covered by atom A. This follows from the fact that the subsimplex is part of the Voronoi polyhedron of atom A. This means that any volume of the subsimplex, which is uncovered by atom A, can not be covered by any other atom of the system.

The formulas for the calculation of the occupied volume and the area of the spherical surface section were first given in Ref. [1]. They are mathematically identical to formulas proposed in Ref. [2], which are represented there in a shorter way. Note, as it was found in our calculations, formula (A8) for the area of the spherical surface section in the pyramid, presented in Ref.[1], is not robust. It is unstable if the subsimplex edge VE tends to zero and at the singular point, where the value of the sphere radius r_C tends to the length of the edge AE. A robust version of this formula is given in Ref. [2].

Here we present the formulas for the occupied volume of the subsimplex, keeping all notations used in Refs.[1,2]. Remember, the subsimplex has one right dihedral angle (between faces ABE and BEV), and therefore right angles between segments AB and BE, and AB and BV, additionally there are right angles between BE and EV, and between AE and EV.

The lengths of the orthogonal edges of the pyramid are x_0, y_0, z_0 , Fig. 4. Thus the length AE is equal to $r_E = \sqrt{x_0^2 + y_0^2}$ and the length AV is $r_V = \sqrt{x_0^2 + y_0^2 + z_0^2}$. The occupied volume depends on which edges of the pyramid are intersected by the surface of the sphere with radius r_C. Thus the following cases are possible:

I: $r_C \le x_0$

$$V_C = \frac{r_C^3}{6}\left(2\theta - \frac{\pi}{2} - a_1\right) \tag{A3}$$

II: $x_0 < r_C \le r_E$

$$V_C = \frac{\theta}{2}\left(r_C^2 x_0 - \frac{x_0^3}{3}\right) - \frac{r_C^3}{6}\left(\frac{\pi}{2} + a_1\right) \tag{A5}$$

III: $r_E < r_C \le r_V$

$$V_C = \left(\frac{r_C^2 x_0}{2} - \frac{x_0^3}{6}\right)\left(\theta - \frac{\pi}{2} + a_2\right) + \frac{r_C^3}{6}(a_3 - a_1) + \frac{x_0 y_0}{6}\sqrt{r_C^2 - r_E^2} \tag{A7}$$

IV: $r_C \ge r_V$

$$V_C = \frac{x_0 y_0 z_0}{6} \tag{A9}$$

where the following notations are used: $\theta = \arctan(z_0/y_0)$, $x_2 = r_C x_0 / r_E$, $y_2 = r_C y_0 / r_E$,

$$a_1 = \arcsin\left[\frac{\left(z_0^2 x_0^2 - y_0^2 r_V^2\right)}{r_E^2\left(y_0^2 + z_0^2\right)}\right], \qquad a_2 = \arcsin\left[\frac{y_0}{\sqrt{r_C^2 - x_0^2}}\right],$$

$$a_3 = \arcsin\left(\frac{x_2^2 - y_2^2 - x_0^2}{r_C^2 - x_0^2}\right).$$

Two subsimplexes are based on each VP face at vertex V. Each vertex V is common to three faces, so it unites six subsimplexes. Typically, point B lies on a face of the VP, and point E is located on a VP edge. For such ordinary configurations the subsimplexes represent the intersection of the VP of A and the DS of its vertex V, Fig.3. In this case these six subsimplexes are combined into a hexahedron with eight corners, which is isomorphic to a cube, and the line segment AV is its spatial diagonal. However, in general they do not form a convex polyhedron, because the inverted subsimplexes, discussed in the 2D section, exist also in 3D. Nevertheless, it is convenient to combine the subsimplexes into a group (sextet) which is affiliated with the pair A - V.

The rule of sign for the summation of three-dimensional subsimplexes was discussed in Refs. [1, 2]. The sign of a subsimplex is determined by the product of three factors: $S_A{\cdot}S_E{\cdot}S_B$. Here a factor S_B is appended to the mentioned factors S_A and S_E for 2D. It reflects the relative positions of point B and the face of the VP, Fig.5.

If point B and the VP face lie on the same side of the edge with the base point E, then the factor S_B for the subsimplex with the base BVE is positive. (For example, in Fig. 5a and Fig. 5c the factor S_B for the subsimplex with the base BVE1 is positive.). Otherwise, S_B is negative, Fig.5b and Fig. 5d. Similarly, for the subsimplex with the base BVE2, the sing is positive in Fig. 5a and Fig.5 b and negative in Fig. 5c and Fig. 5d.

Thus the determination of the subsimplex sign is straightforward: we need to establish the relative position of points on a line (to find the factor S_E), points in a plane relatively to a line (for S_B), and points in space relatively to a plane (for S_A), which can be easily done by elementary analytic geometry.

Note that inverted subsimplexes occur, when the Voronoi vertex V (i.e. the "center" of the DS) is outside its Delaunay simplex. Such a DS is called *open Delaunay simplex* and the corresponding face *open face*. A Delaunay simplex can contain one or two open faces. They usually occur at relatively wide cavities inside atomic systems. If the center is inside, then the Delaunay simplex is called *closed*, and a simplex face which does not separate the body of the simplex and its center is also *closed*. An open face is always adjacent to a closed face of the adjacent simplex [22]. Thus the neighboring simplexes compensate the inverted subsimplexes, see Fig. 2. We will call them the *compensating* Delaunay simplexes.

It is interesting to note that all vertices of the sextet lie on the same circumsphere regardless of whether it forms a convex polyhedron or not. This is evident from the fact that the angles at all vertices B and E are right angles and based on the common diagonal AV (Thales' theorem).

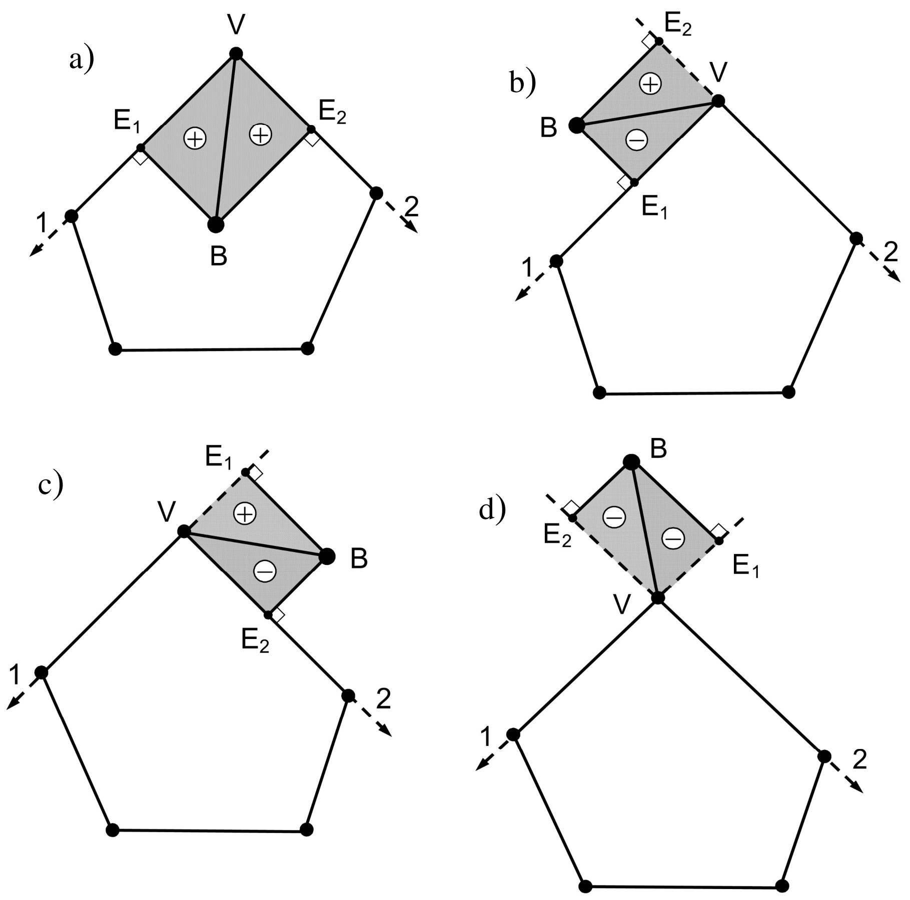


Fig. 5. Choosing the sing of the factor S_B for the calculation of the 3D Voronoi-Delaunay subsimplexes. Possible positions of point B relatively a polyhedron face are shown. Symbols (+) and (-) show the sign of S_B for the subsimplexes with the bases BVE1 and BVE2.

Recall that the perpendiculars from point A to the faces of the VP (segments AB_{ik}) coincide with the edges of the DS. Thus our subsimplexes correspond completely to the subsimplexes defined in Ref. [1]. These authors started from a Delaunay simplex and calculated the subsimplexes related to each corner of the Delaunay simplex. This was reasonable for the analysis of interatomic voids, which are represented as clusters of Delaunay simplexes. In this paper we discuss pairs A-V without an *a priori* reference to Delaunay simplexes, and use the subsimplexes as a general construction element for various structures.

3 Data Structure for the Recording of Subsimplex Volumes

A Voronoi-Delaunay tessellation defines an enumeration of the Voronoi polyhedra (atoms) and Delaunay simplexes. To record their connectivity, a table incidence for the DS and VP is used, where the four numbers of the atoms, which form the *i*-th DS, are recorded in the *i*-th row of the table (array):

$$\begin{array}{c|c|c|c|c|}
\dots & \dots & \dots & \dots & \dots \\
\hline
i & A_1^i & A_2^i & A_3^i & A_4^i \\
\hline
i+1 & A_1^{i+1} & A_2^{i+1} & A_3^{i+1} & A_4^{i+1} \\
\hline
\dots & \dots & \dots & \dots & \dots
\end{array} \qquad (1)$$

Actually each elements of the array (1) represents a pair A-V, mentioned above: a Voronoi polyhedron of atom A_k^i (k =1,2,3,4) and its vertex V, which is the "center" of the Delaunay simplex with the number *i*. Thus, such data structure can be used also to represent sextets of subsimplexes.

Having the Voronoi-Delaunay tessellation of a system, all subsimplexes are determined and their volumes (total, occupied and empty) are calculated according to the formulas (A3) – (A9). The signs of the subsimplexes are established, and the volumes of the sextets are calculated according to the rule of sign for each pair of VP and DS. These data are written in a table similar to (1):

$$\begin{array}{c|c|c|c|c|}
\dots & \dots & \dots & \dots & \dots \\
\hline
i & V_1^i & V_2^i & V_3^i & V_4^i \\
\hline
i+1 & V_1^{i+1} & V_2^{i+1} & V_3^{i+1} & V_4^{i+1} \\
\hline
\dots & \dots & \dots & \dots & \dots
\end{array} \qquad (2)$$

where the elements V_k^i can be the values of the total, the occupied, or the empty volume of the *k*-th sextet of the *i*-th DS, (k =1,2,3,4).

4 Applications

The data structure (2) allows the calculation of the volumes of different structural elements, selected in atomic systems by the Voronoi-Delaunay tessellation. Here we discuss some of them.

4.1 Delaunay Simplex

The total, occupied, or empty volume of the *i*-th simplex is determined by a simple summation of the corresponding elements of the *i*-th row of the array (2). For the *total* volume, such a summation will always give the correct result, even when the simplex is open. Fig. 6 shows a configuration taken from Fig. 2. Here we sum subsimplexes at the atoms A, A' and A". The signs in the figure had been obtained in accordance with the rules of signs for subsimplexes and mark different parts to the total volume of the simplex of vertex V. We see that a non-zero contribution is obtained only for those parts that compose the considered simplex. In this case, the inverted subsimplexes at the atoms A and A" are compensated by the subsimplexes at the atom A'.

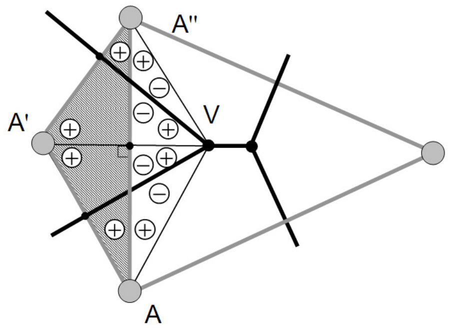


Fig. 6. The summation of the subsimplex volumes over all atoms of any simplex always gives the correct total volume of the simplex, e.g. simplex A, A', A'' of vertex V)

However, a situation as described in Ref. [1] may appear in the calculation of the *occupied* volume. It is known that a Delaunay simplex may include sections of an *extraneous* atom (not from an own vertex of the simplex), Fig. 7a. The volume of the extraneous atom (A) is not considered in the subsimplexes of the neighbor simplex (A', A'', A'''), and therefore can not contribute to the occupied volume of this DS. On the other hand, this volume is taken into account in the subsimplexes of the simplex (A, A'' and A'''), which contains atom A as one of its vertexes, although a part of volume A is outside the simplex. Thus, the method of subsimplexes does not always give the correct occupied volume inside a given simplex. However, for a cluster of these simplexes, such as the entire cavity between the atoms A, A', A'' and A''' we get the correct value of the occupied volume.

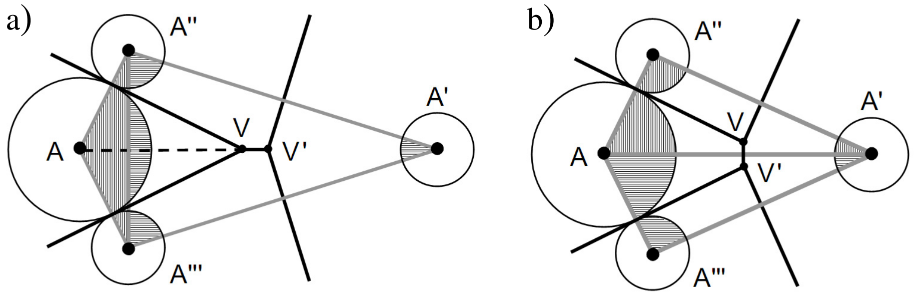


Fig. 7. (a) The penetration of an extraneous atom (A) into the simplex A'A''A''. (b) By a slight shift of the atom A', the Voronoi-Delaunay tessellation will be modified and this peculiarity disappears.

Note that for molecular systems, studied in physics and biology, the appearance of extraneous atoms in the Delaunay simplexes is a rare event. Fig. 7b shows that the non-ordinary atomic configuration of Fig. 7a becomes ordinary after a small shift of atom A' in a direction which reduces the cavity between the atoms (such a shift may happen with high probability in the subsequent step of a molecular dynamics simulation run of a dense molecular system). A modification of the Voronoi-Delaunay tessellation occurs, and the atom A is no longer an extraneous one for the new DS.

4.2 A Local Cavity

A *local cavity* is a cluster of Delaunay simplexes, covering a local void in an atomic system. The outer faces of these Delaunay simplexes are closed and coincide with the closed faces of the neighboring simplexes. In other words, non-ordinary configurations are possible only within the cavity, and each simplex of the cluster has its compensating neighbor. As it was emphasized in Ref. [1], the summation of these subsimplexes gives the correct result as for the total, as well as the occupied and the empty volume of the cavity.

Thus, when calculating the volume of a local cavity, it is enough to sum all rows of arrays (2), which belong to the Delaunay simplexes of a given cavity.

4.3 Voronoi Polyhedron

Unlike for Delaunay simplexes, an extraneous atom cannot affect the occupied (or the empty) volume within a *Voronoi polyhedron*. Therefore, the subsimplexes related to a VP, always give a correct value for the total, as well as the occupied and the empty volume of the VP, see Ref. [2].

The required sextets can be found with the help of array (1). The number of a given atom A, whose VP is considered, is recorded in those rows of array (1), which characterize those simplexes, which have atom A as one of its vertexes. Then the volumes of the required sextets are located in the same places in arrays (2), where the atom A is located in array (1).

4.4 Voronoi Region of a Molecule

The concept of the *Voronoi region of a molecule* in a solution has long been used in molecular biology [3,17]. It is represented by the sum of the Voronoi regions of all atoms of the molecule in solution. Therefore, the calculation of the volume (total, occupied or empty) by the subsimplexes is straightforward, see 4.3.

Note some subtle differences between the concepts of the *occupied (or empty) volume of the Voronoi region of a molecule* and the *molecular volume*, which are used in molecular biology, and can be calculated using the subsimplexes. One usually supposes that the occupied volume of the Voronoi region of a molecule is the van der Waals volume of the molecule. This is true only for the case, when atoms of the molecule do not cross the outer faces of the Voronoi region of molecule. An overlap between an atom of the molecule and an atom of the solvent is shown in Fig. 8. As a result, a part of the atom of the molecule is outside its Voronoi region. Usually only a small fraction of the volume is lost, and the numeric difference between the occupied and the van der Waals volumes is negligible for molecular systems (such overlaps are rare events in molecular systems, due to the strong repulsion between close atoms). We should keep in mind, the van der Waals volume of the molecule is defined as the union of its atoms independently on the solvent.

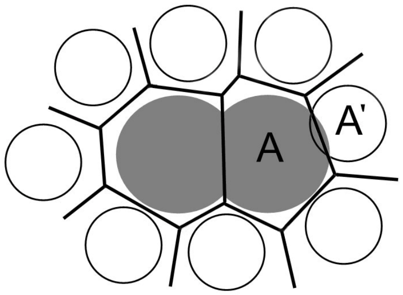


Fig. 8. Occupied volume of the Voronoi region of a solute molecule. A part of an atom A of the molecule lies outside the Voronoi surface of the molecule due to the overlap of this atom with an atom A' of the solvent. This part does not contribute to the occupied Voronoi volume of the solute molecule.

The *molecular volume* is the van der Waals volume of the molecule together with the volume of the inner voids. This is a rather qualitative concept, but a way for a quantitative recording of this value can be adopted. Usually it is calculated with the help of a Connolly surface [4]. The Voronoi-Delaunay technique provides an alternative approach. The molecular volume can be calculated by subtracting the volume of the boundary voids from the total volume of the Voronoi region of the molecule. In section 4.7, we discuss how to estimate the volume of these voids. The molecular volume can be also calculated directly by summing the *occupied* volume of the Voronoi region and the *empty* volume of the Delaunay simplexes, which represent the inner voids of the dissolved molecule (i.e. the DS having vertices that lie only on the atoms of the solute molecule).

4.5 Voronoi Shell

The *Voronoi shell* is defined as an envelope of solvent (water) molecules around the solute molecule. This construction was first proposed in Ref. [23] to allocate the hydration shell. It can also be used to define successive shells of the solvent, surrounding the dissolved macromolecule [18].

Each Voronoi shell is represented by a list of Voronoi polyhedra, forming this shell. Since for each VP its total and occupied volumes are calculated accurately (see 4.3), it only remains to sum all VP, forming the shell, in order to get the desired volume of the Voronoi shell.

4.6 Delaunay Shell

Delaunay shells were proposed for the detection of interatomic voids around the solute molecule [15, 18]. The first Delaunay layer is formed by those Delaunay simplexes, which have vertices of both classes of atoms: from the solute molecule and the solvent. It is a solid shell of Delaunay simplexes around the molecule [18].

The method of subsimplexes gives the correct result for the total volume of the Delaunay layer, since it is the sum of the total volumes of the individual Delaunay simplexes, which are calculated correctly (see section 4.1).

The occupied and the empty volume may have some inaccuracy. This is due to the fact that some DS of the Delaunay shell can have open faces on the outer or inner surface of the Delaunay shell. This means that an extraneous atom can penetrate into the layer, and its volume will be taken into account incorrectly (see section 4.1).

However, from our experience, this discrepancy is small enough for molecular systems. We performed calculations with the molecular models discussed in section 5, to compare the results, obtained by the method of subsimplexes and by a different one, which we had used in our previous works (see e.g. Ref. [7]). This method combines analytical and numerical calculations to find the empty volume of a DS. First, the surroundings of a given DS is examined. If there are no extraneous atoms and at most a triple overlap of spheres, then analytical calculations of the empty volume are performed. Else, the empty volume of such a simplex is calculated numerically. We used this method as reference. It turned out that the maximum difference between the methods does not exceed 1% after the treatment of 1000 independent configurations of the bio-molecule hIAPP in water. A temperature increase can increase the difference, because the overlapping of atoms is magnified. However it is only marginal, about 0.1% in the whole studied temperature interval. This discrepancy can be ignored in our studies of bio-molecules. Note that the calculations by the method of subsimplexes turned out to be two times faster than by our smart combined method.

To compute the occupied or the empty volume of a Delaunay shell, it is sufficient to sum those rows of array (2), which contain the simplexes of a given Delaunay shell.

4.7 Intersection of Voronoi and Delaunay Shells

Of special interest is the empty volume of the intersection of Voronoi and Delaunay shells [19]. The method of subsimplexes allows its calculation, but with some inaccuracy, because the intersection may consist of incomplete parts of Voronoi polyhedra and Delaunay simplexes. In this case, some of the inverted subsimplexes can be uncompensated. Fig. 9 illustrates such a configurations, where two parts of the Voronoi shell fall outside the Delaunay shell. The volumes, which are sticking out will be incorrectly included in the total volume of the intersection.

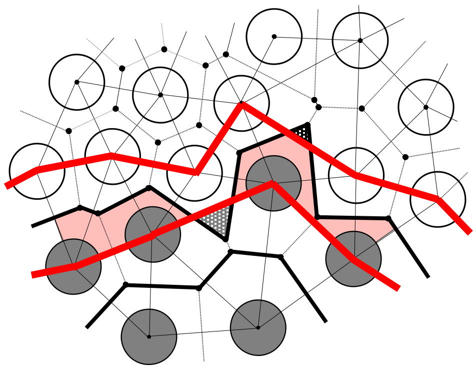


Fig. 9. Illustration of a Voronoi shell, which extends beyond the Delaunay shell. The two areas, , which are sticking out, are dashed. Bold (red) lines border the Delaunay shell. Semi-bold (black) lines show the Voronoi shell. The intersection between the Voronoi and Delaunay shells is shaded (pink). Dark disks are atoms of the solute molecule, light disks are atoms of the solvent.

In addition, there is the problem of extraneous atoms (see 4.1). However, we also think that the resulting inaccuracy is insignificant for our purposes. Unfortunately we can not make a quantitative estimation of the inaccuracy. We do not know any other method to calculate the empty volume of the intersection, which could be applicable for large solute molecules.

Thus, to calculate the empty volume of the intersection, we need to choose those rows in array (2), which correspond to the Delaunay simplexes of the given Delaunay shell, but to sum only those sextets which are related to the atoms (VPs) of the given Voronoi shell.

5 Application to the Calculation of the Volumetric Characteristics of Solutions

The apparent volume of a solute molecule V_{app} (which is the partial molar volume at infinite dilution) consists of the molecular volume (V_M) of the solute molecule, the additional void volume at the boundary between solute molecule and solvent, and the contribution of the solvent due to a local change of the solvent density under the influence of the solute (ΔV). The value V_{app} is measured in physical experiments or calculated independently in computer simulations. The Voronoi-Delaunay method helps to find the components [19].

Fig. 10 illustrates a fragment of a solute molecule in solution. The inner and outer boundaries of the first Delaunay shell (dashed lines) and the outer surface of the Voronoi region of the molecule (solid line) are shown. The Voronoi region of the molecule consists of the molecular volume V_M and a part of the empty volume around the molecule (in Ref. [19] it is marked $V_B{}^M$: the part of the boundary volume, which is assigned to the molecule). Thus $V_{Vor} = V_M + V_B{}^M$. These volumes can be calculated as it is described above in sections 4.4 and 4.7.

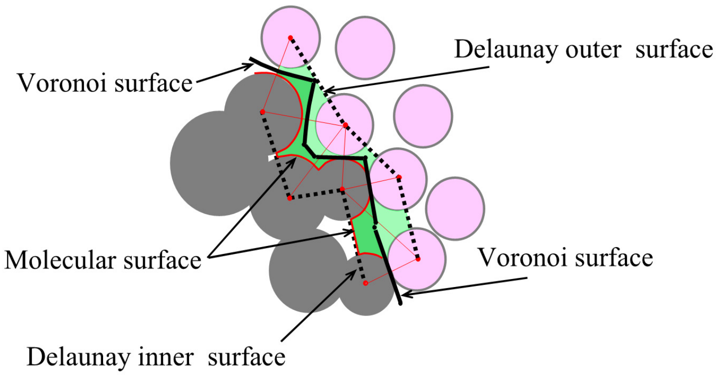


Fig. 10. A fragment of the boundary area between a solute molecule (dark disks) and the solvent (light disks), taken from [19]. The black thick solid line shows the border of the *Voronoi region of the molecule* (V_{Vor}), the so called Voronoi surface. Dotted lines show the inner and outer surfaces of the first *Delaunay shell.* The dark-green area is the boundary empty volume assigned to the solute molecule ($V_B{}^M$). The red thick line over the atomic surfaces and the faces of the Delaunay simplexes represents the surface of the *molecular volume* (V^M) of the solute molecule.

We used the molecular-dynamic models of a single amyloidogenic polypeptide molecule (hIAPP) in aqueous solution, generated in Ref. [14] for different temperatures. 1000 independent configurations, equally spaced over the equilibrated production runs, were used for averaging. Fig. 11 shows the apparent volume and its components for the hIAPP molecule as functions of temperature. V_{app} has been determined in Ref. [15] and the contribution of solvent was calculated as $\Delta V = V_{app} - V_{Vor}$.

It is known, that V_{app} always grows with temperature. However, its components behave in different ways. The molecular volume V_M is practically constant with temperature, but the empty volume $V_B{}^M$ increases. Thus one can see, the apparent volume grows because of the increase of this boundary empty volume and the decrease of the *negative* contribution of the solvent ΔV .

These calculations explain the nature of the thermal expansion coefficient of hIAPP molecule in water. It is related to the surrounding water, but not to conformational or density changes of the molecule itself.

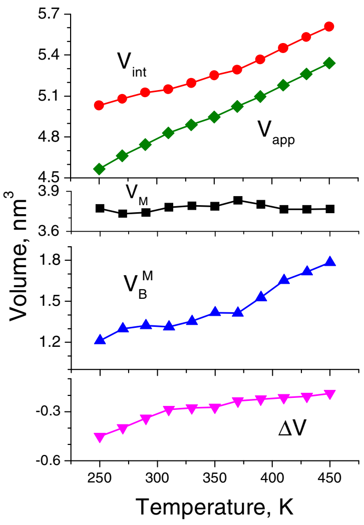


Fig. 11. Apparent volume V_{app} of the hIAPP molecule in water and its components V^M, $V_B{}^M$ and ΔV as functions of temperature

6 Conclusion

This paper describes the application of Voronoi-Delaunay subsimplexes, discussed in Ref. [1, 2], for the calculation of the occupied and empty volumes in molecular systems. A subsimplex is a triangular pyramid constructed at the intersection of a Voronoi polyhedron and a Delaunay simplex. There are analytical formulas, to calculate the occupied volume inside the subsimplex. These formulas and the use of a convenient data structure to record the subsimplexes, enables a fast calculation of the required volumes. Summing up the subsimplexes (using a rule of signs), the occupied (or the empty) volume can be calculated for various structures, composed of Voronoi polyhedra, Delaunay simplexes and their intersections. In some cases the calculated volume might be flawed by slight inaccuracies because of the peculiarities of the Voronoi-Delaunay tessellation in some rarely occurring local packing structures, but its magnitude is not significant for the application to molecular and atomic systems.

To interpret the experimental volumetric data for protein solutions, one needs to know the components of the partial molar volume of the solute molecule and their change with temperature or pressure. Traditionally, the considered components are the volume of the solute molecule itself (molecular volume), the density change in the hydraton water under the influence of the solute (ΔV), and the contribution of additional voids at the border between the solute molecule and the solvent (which relates with the so called thermal volume). The decomposition of the solution into Voronoi and Delaunay shells helps to select corresponding areas in computer models of solutions, and the proposed approach of the subsimplexes enables the calculation of the desired volumes. Using as an example a molecular dynamics model of the protein hIAPP in water, the components of the apparent volume of the molecule were calculated as a function of temperature [19]. This explains the nature of the thermal expansion coefficient of the hIAPP molecule. It originates from the surrounding voids, but not from the molecule itself.

Acknowledgments. Financial support from Alexander von Humboldt foundation and RFFI grant No.12-03-00654 is gratefully acknowledged. We thank M. Andrews and R. Winter for providing us with the trajectories of their hIAPP simulations.

References

1. Sastry, S., Corti, D.S., Debenedetti, P.G., Stillinger, F.H.: Statistical geometry of particle packings. I. Algorithm for exact determination of connectivity, volume, and surface areas of void space in monodisperse and polydisperse sphere packings. Phys. Rev. E 56(5), 5524–5532 (1997)
2. Voloshin, V.P., Anikeenko, A.V., Medvedev, N.N., Geiger, A.: An Algorithm for the Calculation of Volume and Surface of Unions of Spheres. Application for Solvation Shells. In: Proceedings of the 8th International Symposium on Voronoi Diagrams in Science and Engineering, ISVD 2011, pp. 170–176 (2011)
3. Richards, F.M.: Calculation of molecular volumes and areas for structures of known geometry. Methods Enzymol. 115, 440–464 (1985)
4. Connolly, M.L.: Computation of Molecular Volume. J. Am. Chem. Soc. 107, 1118–1124 (1985)
5. Yang, L., Guo, G.Q., Chen, L.Y., Huang, C.L., Ge, T., Chen, D., Liaw, P.K., Saksl, K., Ren, Y., Zeng, Q.S., LaQua, B., Chen, F.G., Jiang, J.Z.: Atomic-Scale Mechanisms of the Glass-Forming Ability in Metallic Glasses. Phys. Rev. Lett. 109, 105502 (2012)
6. Liang, J., Edelsbrunner, J.H., Fu, P., Sudhakar, P., Subramaniam, S.: Analytical shape computation of macromolecules: II. Inaccessible cavities in proteins. Proteins: Struct. Func. Genet. 33, 18–29 (1998)
7. Alinchenko, M.G., Anikeenko, A.V., Medvedev, N.N., Voloshin, V.P., Mezei, M., Jedlovszky, P.: Morphology of voids in molecular systems. A Voronoi-Delaunay analysis of a simulated DMPC membrane. J. Phys. Chem. B 108(49), 19056–19067 (2004)

8. Kim, D., Cho, C.-H., Cho, Y., Ryu, J., Bhak, J., Kim, D.-S.: Pocket extraction on proteins via the Voronoi diagram of spheres. Journal of Molecular Graphics and Modelling 26(7), 1104–1112 (2008)
9. Rémond, S., Gallias, J.L., Mizrahi, A.: Characterization of voids in spherical particle systems by Delaunay empty spheres. Granular Matter 10, 329–334 (2008)
10. Kurzidim, J., Coslovich, D., Kahl, G.: Dynamic arrest of colloids in porous environments: disentangling crowding and confinement. J. Phys.: Condens. Matter 23, 234122 (2011)
11. Chalikian, T.V.: Volumetric Properties of Proteins. Annu. Rev. Biophys. Biomol. Struct. 32, 207–235 (2003)
12. Marchi, M.: Compressibility of Cavities and Biological Water from Voronoi Volumes in Hydrated Proteins. J. Phys. Chem. B 107, 6598–6602 (2003)
13. Mitra, L., Smolin, N., Ravindra, R., Royer, C., Winter, R.: Pressure perturbation calorimetric studies of the solvation properties and the thermal unfolding of proteins in solution—experiments and theoretical interpretation. Phys. Chem. Chem. Phys. 8, 1249–1265 (2006)
14. Brovchenko, I., Andrews, M.N., Oleinikova, A.: Volumetric properties of human islet amyloid polypeptide in liquid water. Phys. Chem. Chem. Phys. 12, 4233–4238 (2010)
15. Voloshin, V.P., Medvedev, N.N., Andrews, M.N., Burri, R.R., Winter, R., Geiger, A.: Volumetric Properties of Hydrated Peptides: Voronoi-Delaunay Analysis of Molecular Simulation Runs. J. Phys. Chem. B 115(48), 14217–14228 (2011)
16. Cazals, F., Kanhere, H., Loriot, S.: Computing the volume of a union of balls: a certified algorithm. ACM Transactions on Mathematical Software 38(1), 3 (2011)
17. Procacci, P., Scateni, R.: A General Algorithm for Computing Voronoi Volumes: Application to the Hydrated Crystal of Myoglobin. International Journal of Quantum Chemistry 42, 1515–1528 (1992)
18. Kim, A.V., Voloshin, V.P., Medvedev, N.N., Geiger, A.: Decomposition of a Protein Solution into Voronoi Shells and Delaunay Layers: Calculation of the Volumetric Properties. In: Gavrilova, M.L., Tan, C.J.K., Kalantari, B. (eds.) Transactions on Computational Science XX. LNCS, vol. 8110, pp. 56–71. Springer, Heidelberg (2013)
19. Voloshin, V.P., Kim, A.V., Medvedev, N.N., Winter, R., Geiger, A.: Calculation of the volumetric charateristics of macromolecules in solution by the Voronoi-Delaunay technique. J Phys. Chem. B. (submitted, 2014)
20. Mezei, M.: Modified proximity criteria for the analysis of the solvation of a polyfunctional solute. Molecular Simulation 1(5), 327–332 (1988)
21. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM J. Comput. 16, 78–96 (1987)
22. Medvedev, N.N.: Voronoi-Delaunay method for non-crystalline structures. SB Russian Academy of Science, Novosibirsk (2000) (in Russian)
23. David, E.E., David, C.W.: Voronoi Polyhedra as a Tool for Studying Solvation Structure. J. Chem. Phys. 76, 4611 (1982)
24. Andrews, M.N., Winter, R.: Comparing the Structural Properties of Human and Rat Islet Amyloid Polypeptide by MD Computer Simulations. Biophys. Chem. 156, 43–50 (2011)

Author Index