Stony Brook University

Structure Analysis to Large Size Electronic Systems and Its Application for Optimized Electronic Design

A Dissertation Presented

by

Yang Zhao

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

August 2008

Stony Brook University

The Graduate School

Yang Zhao

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Dr. Alex Doboli, Dissertation Advisor

Associate Professor of Electrical and Computer Engineering

Dr. Thomas Robertazzi, Chairperson of Defense

Professor of Electrical and Computer Engineering

Dr. Monica Fernandez-Bugallo

Assistant Professor of Electrical and Computer Engineering

Dr. Edward H. Currie, Outside Member

Chief Information Officer of Tritium Technologies, Inc.

This dissertation is accepted by the Graduate School

Lawrence Martin

Dean of the Graduate School

Abstract of the Dissertation

Structure Analysis to Large Size Electronic Systems and Its Application for Optimized Electronic Design

by

Yang Zhao

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2008

The discovery of small-world and scale-free properties of many nature, society and artificial complex networks has stimulated a great deal of interest in studying the underlying organizing principles of various complex networks. This study presents a methodology that helps to understand large size electronic circuits' topologies before the physical realization of the system.

My data show that large size electronic circuits' netlists have broad-scale patterns and large clustering coefficients. Both attributes are very

different from random graphs. Furthermore, I introduce a model to explain the large clustering coefficient of electronic circuits' netlists. By applying this structure analysis, I propose an analytical algorithm for general floorplan and placement in the physical design of VLSI (Very Large Scale Integration). The new algorithm implements a partitioning based method in a top-down hierarchical way, which uses $hMETIS$ as a hypergraph and circuit partitioning tool. The floorplanning blocks are represented in integer programming formulation and accurately placed using a non-linear solver named $SNOPT$. This new method decreases the CPU time when taking electronic circuits with large number of blocks.

In addition to this electronic circuits' floorplan and placement experiment, this study offers an approach to performance predictive collaborative control of UAVs (Unmanned Autonomous Vehicles) operating in environments with fixed and pop-up targets. I find an integer linear programming based solution for assigning and scheduling the fixed targets to UAVs and for computing the slack time intervals used for collaborative actions.

The common topic for both studies is based on problem solving technique. The algorithm of electronic circuits' floorplan and placement is realized by INLP (Integer Non-Linear Programming), and the tasks' scheduling and assigning for different UAVs are constructed by ILP (Integer Linear Programming).

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor, Dr. Alex Doboli, for his guidance and help throughout my Ph.D study. He has taught and helped me in my research with great patience and kindness. From him, I have learned the manner of a devoted and energetic scientist and a knowledgeable and caring teacher.

I want to thank my committee members: Dr. Thomas Robertazzi, Dr. Monica Fernandez-Bugallo, and Dr. Edward H. Currie, for their time and effort in reviewing this work, and attending my dissertation defense. Thanks to Dr. Wendy Tang and Dr. Hui Zhang for being my preliminary defense committee members. Their advice is greatly appreciated.

I wish to thank Rohit Pai for his initial work on floorplanning algorithm, it is the ancestor of *Hercules* now. I am grateful to Hui Zhang, Ying Wei and Sankalp Kallakuri for their inspiring suggestions and talks. Their thoughts and work directly contributed to my research life.

I am deeply indebted to my friends Ming Ma, Zejie Zhang, uncle Qidong Cao, Lihua Yu, Anfei Li. I will never forget their invaluable support and help at the most needed time in my life.

Thanks to all the colleagues, classmates, and friends who made this dissertation possible and a joyful experience for me: Hua Tang, Nattawut Thepayasuwan, Bhaskar Mukherjee, Yulei Weng, Lei Wang, Junling Zhou, Pengbo Sun, Jing Gao, Meng Wang, Cristian Ferent and Varun Subramanian.

I am thankful for my wife Peihua Yuan. She gives me hundred percent support no matter what, she stands beside me and gives me a swift kick when I need it. She loves me and forgives me. She is my best friend and a truly soul mate. I am so fortunate and grateful, thanks for the happiness and sadness we shared, sharing and will share.

Finally I wish to express my deepest gratitude for the constant support, understanding and love from my parents. My dear mother Chen Yi and my adored father Weiren Zhao. I am also very grateful to my parents in law, aunt, uncle and all my cousins, specially to Liang Pan for his friendship and help. This dissertation is also a gift to my grandfather, Chung-Yao Chao, Suzhi Yi and grandmother Yibing Huang for their deep love.

Chapter 1

Introduction

1.1 Thesis Motivation

Complex networks are capable of describing a wide range of systems in nature and society, frequently cited examples include the World Wide Web (a network of routers and computers connected by physical links), the brain (a network of neurons), an organization (a network of people). While traditionally these systems have been modelled as random graphs, it is increasingly recognized that the topology and evolution of real networks are governed by robust organizing principles [6].

For over a century, modelling of physical as well as non-physical systems and processes has been performed under an implicit assumption that the interaction patterns among the individuals of the underlying system or process can be embedded onto a regular structure such as a Euclidean lattice. In late 1950s, two mathematicians, Erdös and Rényi (ER), made a breakthrough in classical mathematical graph theory. They described a network with a complex topology by a random graph [2]. Their work has become the foundation of the random network theory, and was followed by continuous studies over the next 40 years and continues even today. Although intuition clearly indicates that many real life complex networks are neither completely regu-

lar nor completely random, the ER random graph model has proven to be the only sensible and rigorous approach for most scientists thinking about complex networks for nearly half of a century. This is due to the absence of super computational power and detailed topological information about very large scale real world networks.

In the past few yeas, the computerization of data acquisition and the availability of high computing power have led to the emergence of huge databases on various real networks of complex topology. In this endeavor, two significant recent discoveries are the small-world effect and the scale-free feature of most complex networks.

In 1998, in order to describe the transition from regular lattice to a random graph, Watts and Strogatz (WS) introduced the concept of small-world network [13]. A interesting popular manifestation of the "small-world effect" is the so-called "six degrees of separation" principle, suggested by a social psychologist, Milgram , in the late 1960s [3]. A prominent common feature of the ER random graph and the WS small-world model is that the connectivity distribution of a network peaks at an average value and decays exponentially. Such networks are called "exponential networks" or "homogeneous networks," because each node has about the same number of link connections.

Another significant discovery in the field of complex networks is the observation that many large scale complex networks are scale-free, that is, their connectivity distributions are in a power-law form that is independent of the network scale [4] [5]. Unlike an exponential network, a scale-free network is inhomogeneous in nature: most nodes have very few link connections and yet a few nodes have many connections. The discovery of the small-world effect and scale-free feature of complex networks has led to dramatic advances in the field of complex networks theory in the past few years.

Electronic circuits can be viewed as networks in which vertices (or nodes) are electronic components (e.g. logic gates in digital circuits and resistors, capacitors,

diodes and so on in analogic circuits) and connections (or edges) are wires in a broad sense. Therefore, large size electronic circuit topologies can be seen as a complex networks. In the system-on-chip (SoC) field, the chips are getting more and more compact and complex. As a result, design and circuit optimization becomes quite difficult. In this thesis, a basic topology structure inside the electronic circuit netlists is implemented to optimize the floorplan and placement process in VLSI physical design. In Chapter 4, a similar problem solving technique utilizing an integer linear programming algorithm is used to study the performance predictive collaborative control of Unmanned Autonomous Vehicles (UAV) in environments with fixed and pop-up targets.

1.2 Goals and Contributions

A review of the literature confirms that there has been little work [7] done on the electronic circuits with complex network theory. The work presented in this thesis is just a preliminary step, and focus on large size electronic circuit netlists. Two circuit topology characteristics have been found in the broad-scale patterns and large clustering coefficient. These attributes are shown to be very different from random graphs, usually assumed for representing circuit topologies. Furthermore, a Triad formation (TF) model is introduced to explain the coexistence of broad-scale patterns and large clustering coefficients.

In considering the electronic circuit topologies structure, a hierarchical mixed integer programming based algorithm for floorplanning in VLSI design is developed. The method is novel in that:

1. A top-down level hierarchical floorplanning methodology is used instead of the traditional flat level structure. This methodology significantly reduces the com-

putational time when handling the large size electronic circuits compared to other methods.

2. It uses a mixed integer programming-based algorithm to represent the flooplan solutions topology. This method uses binary variable pairs to represent any two blocks relative positions. It can be simply merged and optimized with simulated annealing's different perturb moves.

3. The use of a non-linear solver package called SNOPT (which is also based on mixed integer nonlinear programming [MINLP]) reduces the number of constraint variables in the floorplanning problem.

4. The final MCNC benchmark example results show that Hercules (the name of the developed algorithm) has some advantages when solving large size electronic circuit netlist floorplans. It is noted that further improvement may be possible if more structure information is used in the partition step before the top-down framework.

5. The non-linear solver package SNOPT has different implementations depending on the type of problem to be solved. Chapter 4 describes an integer linear programming method for task assignment and scheduling in UAVs' application. Mixed-integer linear programming (MILP) is a powerful optimization method that extends continuous linear programming to include binary or integer decision variables.[8]-[12] These 0/1 integer variables can be further used to model logical constraints such as collision avoidance rules, tasks scheduling constraints and flexibility constraints for UAVs's collaboration.

1.3 Thesis Organization

Chapter 2 details how to find Broad-Scale patterns and large clustering coefficient in a large electronic circuit. A new hierarchical mixed integer programming based algorithm for floorplanning is analyzed in Chapter 3. In Chapter 4 an integer linear programming method is implemented for task assignment and scheduling for collaborating UAV (Unmanned Autonomous Vehicles). Chapter 5 presents conclusions based on the work embodied in this thesis and suggests area for future study.

Chapter 2

Finding Broad-Scale Patterns in Large Size Electronic Circuit

2.1 Introduction

It is a common practice to represent systems with complex topologies and unknown organizing principles as random graphs. However, there is strong evidence that real networks and systems are complex and well structured, thus not random. For example, the power-law degree distribution (scale-free network) was observed for the World Wide Web by Barabási and Albert in 1999 [4]. Later, this power-law distribution feature was found in many other large-scale network topologies, including social and biological networks. These observations are quite intriguing and may apply to electronic design automation (EDA). Traditionally EDA, very large circuit and system netlists are assumed to be random networks. Then, steps like circuit partitioning, placement and routing operate are using the random network assumption, and without considering any details about the netlist structure. However, as we explained in Section 5, knowing the structural properties of netlists can improve the quality of design automation tools.

In this chapter we analyze and contrast electronic circuit topologies with scale-

free graph topologies. The analysis of electronic circuit topologies has been done by methods analogous to those used to analyze scale-free graphs [6]. We present evidence for the existence of broad-scale patterns in electronic circuits as well as an analysis of their degree distributions and clustering coefficients. Broad-scale networks [7]define power-law degree distributions within a certain range. $P(k) \sim k^{-\gamma} f(k/k^*)$, where k* gives the cutoff degree.

This chapter has the following structure. Section 2 summarizes related work of scale-free network and small-world network. Section 3 and 4 first analyze the large size electronic circuit properties, including degree distribution and clustering coefficients, and then introduces a model to explain the observed large clustering coefficient. Section 5 discusses how the properties of large size electronic circuit netlists can help placement and routing in the physical design problem and conclusions are presented.

2.2 Related Work

In the past few years, following the discovery of small-world and scale-free networks, the underlying organizing principles of various natural and artificial complex systems have been studied.

2.2.1 Scale-free networks

Networks with a power-law degree distribution are called scale-free[4]. For a large number of networks, including the World-Wide Web[18], Internet[19], metabolic and protein networks[20, 21], the degree distribution has a power-law tail with an exponent γ(see equation 2.1).

$$P(k) \quad \sim \quad k^{-\gamma} \tag{2.1}$$

where k is the degree of a given node, representing the number of edges connecting it with other nodes and $P(k)$ is defined as the (normalized) probability of nodes having k edges[6]. In addition to the power-law phenomena observed in many complex system, Barabási and Albert[4] further introduced the Barabási-Albert model(BA model), which can generate a network with a power-law degree distribution. There are two basic mechanisms in the BA model, one is *Node Growth* and the other is *Preferential Attachment* [4, 5, 6].

(1) *Node Growth:* Starting with a small number(m_0) of nodes, at every timestep a new node with m($\leq m_0$) edges is added that links it to m different nodes already present in the system.

(2) *Preferential Attachment:* When choosing the nodes to which the new node connects, it is assumed that the probability II that the new node will be connected to node i depends on the degree k_i of node i, such that

$$II(k_i) \quad = \quad \frac{k_i}{\sum_j k_j} \tag{2.2}$$

After t time steps, the PA rule results in a network with $N = t + m_0$ nodes and mt edges. The PA rule assumes that the likelihood of receiving new edges increases with the node's degree. This assumption involves two hypotheses: first, that $II(k)$ depends on k, in contrast to random graphs in which $II(k) = p$, and second, that the functional form of $II(k)$ is linear in k[6]. Numerical simulations indicate that this network evolves into a scale-free state with *P(k)* following a power-law with an

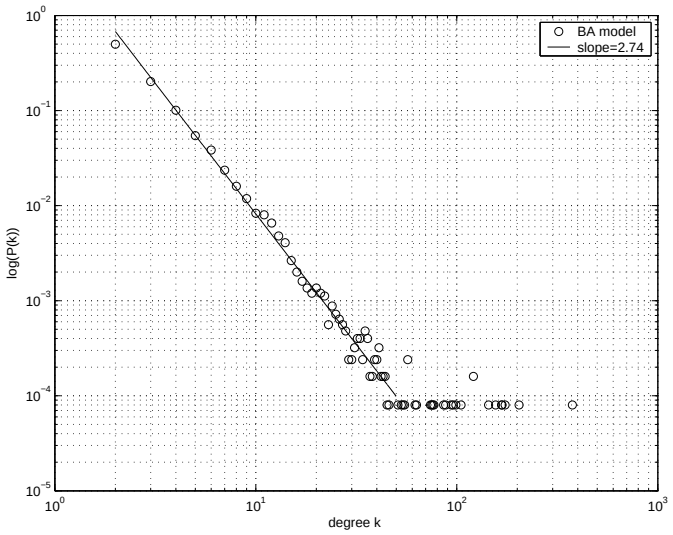

Figure 2.1: Simulation Results for Scale-free networks, with $N = m_0 + t = 12506, m_0 = 3, m = 2$(circles). The slope of the solid line is $\gamma = 2.74$.

exponent $\gamma = 3$ (Fig. 2.1). The bigger the number of nodes, the closer γ nears to 3.

2.2.2 Small-world networks

Between completely regular and completely random systems, Watts and Strogatz [13] found some systems which are highly clustered, but have small characteristic path lengths. These networks are called small-world. In the real world, social networks have the typical small-world characteristic, where most people are friends with their immediate neighbors. For example, classmates in the same school or colleagues in the same office. On the other hand, many people have a few friends who are far away, such as pen pals in other countries.

In order to represent small-world networks, Watts and Strogatz introduced an interesting model, referred to as WS small-world model [13, 14]. The WS model can be generated as follows:

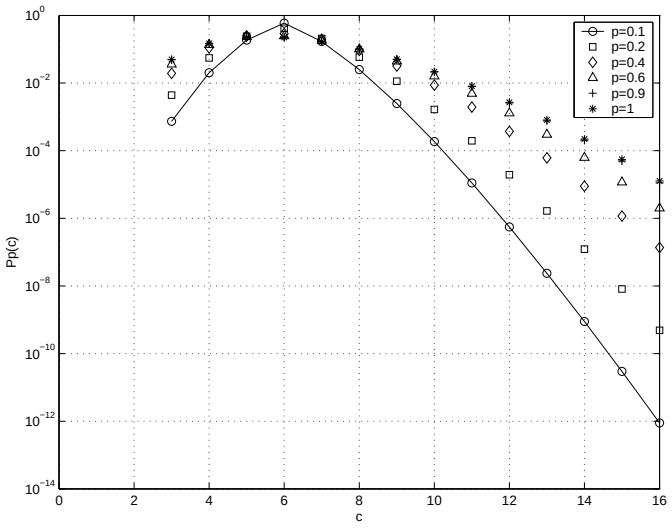

Figure 2.2: Probability distributions of the connectivity c for $k = 3$ and various values of p [14]

Step 1: *Start with regular network:* Begin with a one-dimensional N vertex ring, each vertex being connected to its $2k$ nearest neighbors.

Step 2: *Randomized rewiring:* Randomly rewire each edge of the network with probability p. Long range connections will be introduced in this step.

The probability distribution of the connectivity c should satisfy equation 2.3. The degree distribution of a sample small-world network is shown in Fig. 2.2

$$P_p(c) = \sum_{n=0}^{min(c-k,k)} \binom{k}{n} (1-p)^n p^{k-n} \frac{(kp)^{c-k-n}}{(c-k-n)!} exp(-pk), c \geq k \tag{2.3}$$

Comparing the BA model's degree distribution with the WS model's degree distribution, we can find the distinct differences. For WS model(Fig. 2.2), the connectivity distribution of the network is homogenous, which peaks at an average value($< c >= 6$)

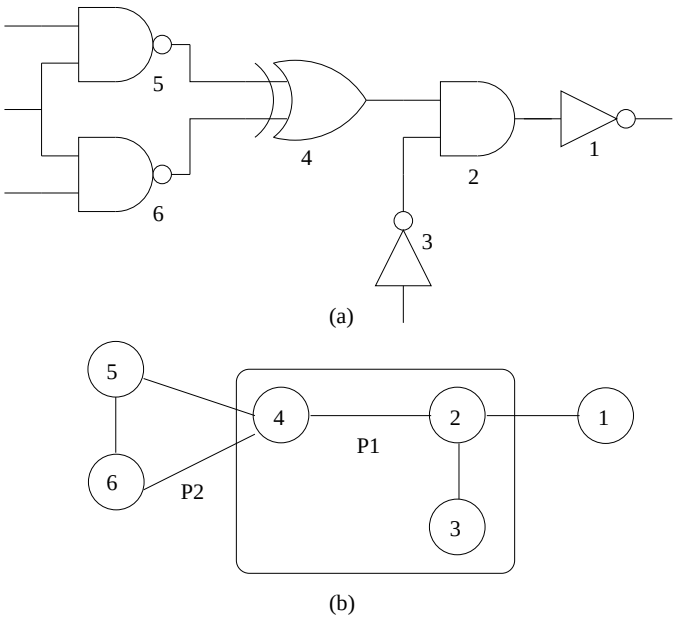

Figure 2.3: (a) An electronic circuit (b) Its corresponding graph

and decays exponentially. For the BA model(Fig. 2.1) of a scale-free network, their connectivity distributions have a power-law form, and the average degree $< k >$ can not be summarized from the degree distribution graph.

2.3 Large Size Electronic Circuit Degree Distribution

Electronic circuits can be viewed as networks in which vertices (or nodes) are electronic components (Fig. 2.3).

In the present case reference is made to the IBM-PLACE benchmark circuits in Generic Hypergraph formats. Inside a netlist file, there are thousands of nets and nodes. For example, $ibm01$ netlist has 12282 nodes and 11507 nets. Therefore, they are large size electronic circuit netlists. The nodes correspond to the logic gates in

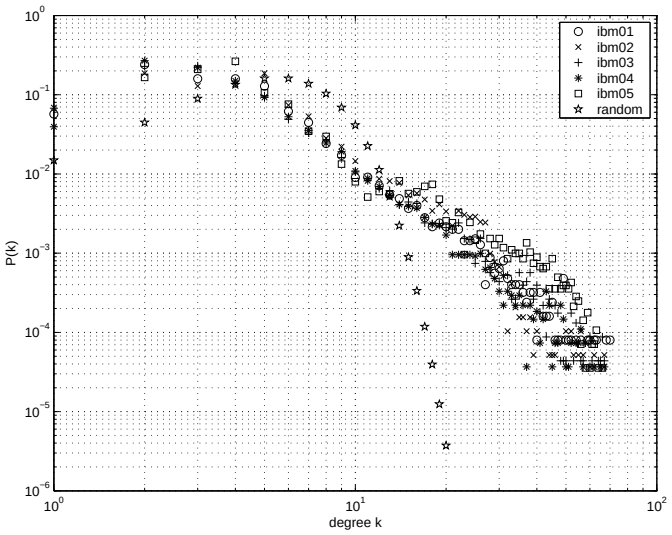

Figure 2.4: Degree Distribution of $ibm01 - 05$ and random graph

a digital circuit. The edges connecting any two nodes represent the connectivity of the circuits netlists. Using the formalism of graph theory, the whole circuit can be described in terms of a flat graph G, consisting of a vertex set $V(G)$ and an edge set $E(G)$[16].

The analysis of a network's degree distribution can help determine which kind of networks large electronic netlists belong to. The degree distribution for the five large electronic circuit netlists($ibm01 - ibm05$) are shown in Figure 2.4. For comparison, a random graph's distribution is included on the same scale. This random graph's degree distribution satisfies Poisson distribution ($N = 10000, < k >= 6$) which shows the large scale electronic circuit netlists are not random graphs and illustrates that the degree distribution has power-law curve within certain range and having a bottom cut-off point at $k^* \approx 3, 4$. The power-law curve becomes indiscernible after $k > 30$, because there are always a few nodes that have large degree, but the number of these nodes is small, approximately $1 - 2\%$ of the total number of nodes. The average degree has been also calculated as $< k > \approx 4 - 6$.

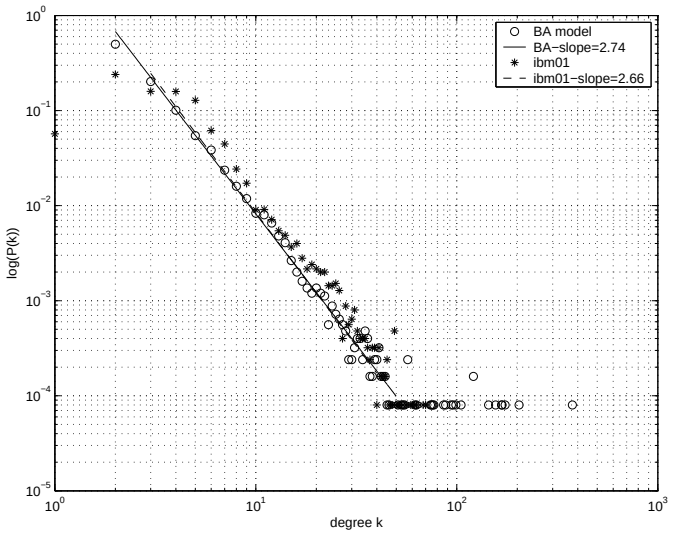

Figure 2.5: Degree Distribution of $ibm01$ and BA model($m_0 = 3, m = 2$)

The Figure 2.5 depicts the degree distribution comparison between $ibm01$ and BA model. It presents the expected distribution for a corresponding BA model with the same number of nodes($N = 12506$) for $ibm01$. The results show that both the curves and γ values are quite close. Within the power-law range, curve fitting gives an exponent $\gamma \approx 2.66$, the corresponding BA model's $\gamma \approx 2.74$.

For different scale-free networks, numerical values of the exponent γ for various systems are diverse, but most of them are in the range of $2 < \gamma \leq 3$ [15]. Compared with Table 2.1, all these large size electronic circuit netlists γ comply with this condition. From a degree distribution point of view, it can be concluded that large size electronic circuits have broad-scale patterns.

Table 2.1: $ibm01-05$ curve fitting γ

IBM files	Number of Nodes	Range of k	Node Percentage[%]	$-\gamma$
ibm01	12506	$k \in [4, 30]$	66%	2.66
ibm02	19342	$k \in [4, 30]$	72%	2.30
ibm03	22853	$k \in [3, 30]$	88%	2.44
ibm04	27220	$k \in [3, 30]$	92%	2.63
ibm05	28146	$k \in [4, 30]$	82%	2.42

2.4 Large Size Electronic Circuit Clustering Coefficient

2.4.1 Definition of clustering coefficient

The clustering coefficient C measures the cliquishness of a network, which is defined as follows[13]. Given a selected node i in the network, having k_i edges which connect it to k_i other nodes; then at most $k_i(k_i-1)/2$ edges can exist between them. The ratio between the number E_i of edges that actually exist between these k_i nodes and the total number $k_i(k_i-1)/2$ gives the value of the clustering coefficient of node i.

$$C_i = \frac{2E_i}{k_i(k_i-1)} \tag{2.4}$$

The clustering coefficient of the whole network C is the average of all individual C_i's.

$$C = < C_i > = \frac{\sum_{i=1}^{N} C_i}{N} \tag{2.5}$$

Table 2.2: $ibm01-05$ clustering coefficient

IBM files	Average Degree $< k >$	Clustering Coefficient C	C_{rand} $\times 10^{-4}$	C/C_{rand}
ibm01	4.8	0.16	3.96	412
ibm02	5.5	0.15	2.91	520
ibm03	4.6	0.13	2.10	617
ibm04	4.4	0.10	1.69	568
ibm05	5.8	0.11	2.05	542

2.4.2 Random graph and BA model's clustering coefficient

In a random graph, since the edges are distributed randomly, the clustering coefficient is $C_{rand} = p = \frac{<k>}{N}$ [6]. Small-world networks, in addition to a short average path length, always have a relatively high clustering coefficient. Comparing the BA model of scale-free network with a random network, the BA model's clustering coefficient is about five times higher [6]. The large size electronic circuit netlists clustering coefficient has been analyzed at Table 2.2.

2.4.3 TF model's clustering coefficient

The clustering coefficient of large size electronic circuit netlists is so big that the BA model can't explain it. In order to represent large size electronic circuit netlists, a new model is proposed which has both the power-law degree distribution and the high clustering.

Reviewing the BA model of the scale-free network, it is noted that there is an important step called *Preferential Attachment (PA)*. In this step, the newly introduced node is always attached to the existing node with large degree. Through this process,

the large degree node becomes larger, the small degree node becomes smaller. Finally, an examination of the network shows that there are a few hubs inside it, representing the large degree nodes, and all the other nodes become too scattered to compose clustering. This is the reason that the BA model's clustering coefficient is not big enough as the real world scale-free network. Petter and Beom [17] offered a *Triad formation (TF)* method to increase the scale-free network's clustering coefficient. It is well known that a triad is the simplest graph composition which is fully connected, and $C = 1$. Their idea was to introduce more triads to increase the final network's clustering coefficient, so they modified the BA algorithm by adding an additional step.

If an edge between v and w was added in the previous PA step, then add one more edge from v to a randomly chosen neighbor of w to form a triad. If there remains no pair to connect, do a PA step instead.

In order to make the clustering coefficient tunable, the TF step is performed with the probability P_t and the PA step with the probability $1 - P_t$. By using this method, the corresponding $TF01 - TF03$ models (with same number of nodes) for $ibm01 - ibm03$ are obtained. Obviously, the clustering coefficient increases from the original BA model. TF model's clustering coefficient is quite close to their actual cluster coefficient as seen from Table 2.3. The $TF01$ model's degree distribution is further achieved as shown in Fig 2.6. Not only the cluster coefficient has increased dramatically from the original BA model, but also the power-law property has been preserved.

2.5 Discussion

The physical design(circuit layout problem) is normally divided into system partitioning, floorplanning, placement, and routing. Each of the steps must be performed

Table 2.3: $ibm01-03$ and TF clustering coefficient

IBM files TF model	Clustering Coefficient C	C_{BA} $\times 10^{-2}$	C/C_{BA}
ibm01	0.16	0.25	64
ibm02	0.15	0.26	57
ibm03	0.13	0.24	55
$TF01(P_t = 0.2)$	0.15	0.25	60
$TF02(P_t = 0.1)$	0.14	0.26	54
$TF03(P_t = 0.05)$	0.13	0.24	55

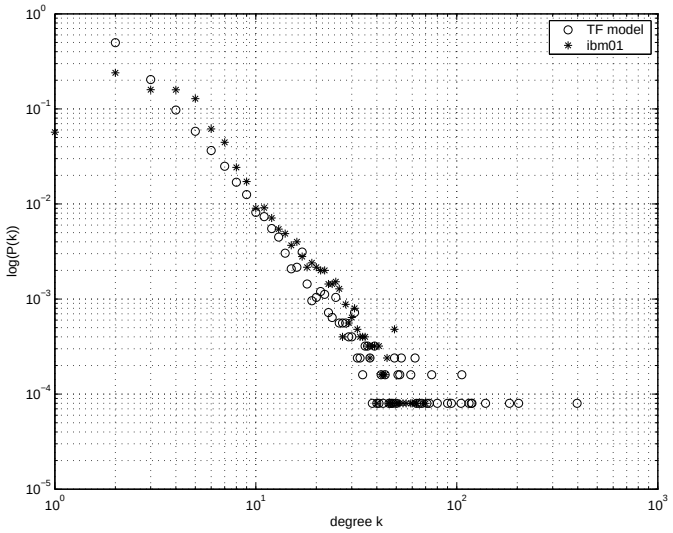

Figure 2.6: Degree Distribution of $ibm01$ and TF model($m_0 = 3, m = 2, P_t = 0.2$)

and each depends on the previous step. However, the trend is toward completing these steps in a parallel and iterating, rather than in a sequential manner. If some of the circuit topologies are known before the physical design process, they can be used to facilitate floorplanning and placement steps.

The broad scale-free degree distribution shows that major topological differences exist between random networks and large size electronic circuit netlists. For the random networks, most nodes have approximately the same number of links, $k \approx < k >$, the exponential decay of $P(k)$ guaranteeing the absence of nodes with significantly more links than $< k >$. In contrast, the power-law distribution of the most nodes in large size electronic circuit netlists implies that the structure has two different kinds of nodes. There are numerous nodes with only a few links, but a few nodes have a very large number of degree or links. These nodes can been seen as hubs inside a circuit. When system partitioning is imposed, considering these hubs can result in these hubs being distributed in different partition parts. During floorplanning, when the sub-circuits(blocks) locations are assigned, if there is more than one hub in the sub-circuits(blocks), these hubs should be kept close to each other. By doing this, the highly connected blocks are kept physically close to each other which minimizes interconnection area and wirelength in the next placement and routing steps.

The large clustering coefficient of large size electronic circuits netlists implies there may be some patterns of interconnections occurring in complex electronic circuits. The patterns or building blocks are found in many fields of science, such as biochemistry, neurobiology, ecology, and engineering [22]. The patterns of large size electronic circuits netlists can help in locating the logic cell within the flexible blocks in the placement step. For a special case, if there are many cliques (fully connected graph) inside the blocks, then during the placement step, these cliques may become recognizable as basic cells and it won't matter what nodes inside a cell, because all of the nodes wirelengths are the same in a clique.

2.6 Conclusion

This chapter explains that large size electronic circuit netlists have broad scale-free patterns by offering their power-law degree distribution in a range and comparing it with BA model. Also it was found that the circuit's clustering coefficient is too large for BA model. The introduced TF model delivers a proper answer to the large clustering coefficient. The results presented may save tedious optimization work during placement and routing, and help to provide new ways of generating a large circuit testbench, which are important objectives for future work.

Chapter 3

A Hierarchical Mixed Integer Programming Based Algorithm for Floorplanning in VLSI Design

3.1 Introduction

The chapter focuses on the floorplan and placement design problem of very large scale integrated (VLSI) circuits. At these particular design steps, the VLSI circuit is seen as a set of rectangular blocks (modules) on a two dimensional surface such that no two blocks overlap, while optimizing certain objectives (minimize chip size and total wire length). Floorplanning helps solve such problems. This is related to placement, since the blocks shape and pin positions on the periphery of chip components are fixed. Floorplanning becomes more and more important for a hierarchical layout style used in very deep sub-micron design. The topic has been studied [26] for more than a decade. The rectangular block in VLSI circuit (netlists) consists of hundreds or thousands cells, which may performs logical or arithmetic operations such as AND,NOR and flip-flops, and the size of each block is predetermined. Blocks are grouped into two types according to their shape and flexibility, viz., hard and soft blocks. Hard blocks are rigid and have fixed shapes, while soft blocks have widths and heights that are free to change provided that their aspect ratios remain within

a certain range. In the following discussion, the focus is on soft blocks so that any topology can be handled properly.

One of the early works in the flooplanning field was done by [28] and [29]. Wong and Liu [29] propose a normalized Polish expression to represent slicing floorplans and use a simulated annealing (SA) algorithm to obtain a floorplan. The novelty of their methodology is the simultaneous consideration of the shape and interconnect information. But SA fails as the number of blocks increases. The methodology developed in this thesis improves on this aspect by using hMETIS as partition tool to reduce a big netlist into levels of small netlists. The top-down framework make the SA algorithm more efficient. Young *et al.* [39], [40] extend Wong's algorithm to handle cases which certain blocks should be adjacent to specific boundaries of a chip. To handle nonslicing floorplans, Murata *et al.* [32] and Murata and Kuh [33] offer methods based on the sequence-pair topology presentation, while Nakatake *et al.* [35] and Kang *et al.* suggest methods based on the bound-sliceline-grid (BSG). In these methods, a floorplan topology is represented by a pair of sequences of block indexes, and for a BSG, which is a plane dissected into several rectangles by horizontal or vertical line segments.

Temo Chan [30] and Pinhong Chen [31] both uses linear programming to solve the floorplan sizing problem. The sizing problem is modeled as a convex optimization problem. Based on sequence-pair representation [32] [33] and a linear programming method [31], Kim*et al.* [34] suggested that a floorplan design problem can be split into two sub-problems namely the topology generation and floorplanning area minimization (FAMP). The topology generation step helps to fix the relative positions of blocks so that the wire length can be minimized. In FAMP, blocks positions (x,y coordinates) and dimensions (width and height) are further determined such that the total area is minimized. However, splitting the problem into two stages may diminish the solution quality since the two stages are done sequentially and relatively

independently.

A preferred approach is to focused on the floorplanning problem in which soft blocks are to be placed within a chip for the objective of minimizing chip size and total wire length. The work carried out in this thesis developed a top-down level framework, and implemented a partition tool hMETIS first to reduce a big problem, such as that of most previous studies. In previously published work, other authors treated the floorplanning problem in a flat level, with the result that the computation time increased as the size of problem (number of blocks) was increased. The method proposed in this thesis offers substantial benefits when the floorplanning problems are large in terms of size. In this method the mixed integer programming formulation (MIP) introduced by Sutanthavibul *et al.* [37] is used to represent the topology of floorplans instead of sequence-pair which were commonly used in previous papers. For reasons that will be discussed, the MIP model is more concise and clearer when describing a small number of blocks relatively positioned. Blocks width-height ratio and exact positions are solved by a non-linear solver, called SNOPT [25]. This also makes a difference with the linear programming approximation methods [30] [31] [34] mentioned before. Because of the particular partition-based method used for this work, terminal propagation (TP) step [53] is reinforced in the programming to further decrease the wire length. The SA algorithm was also chosen to find a best MIP that gives the best floorplan: one with the minimum total wire length and chip size.

The rest of this chapter is organized as follows. In Section 2, the definitions used in this thesis are presented, and a description of the floorplanning problem is provided. Section 3 introduces the Top-Down floorplanning methodology. A Mixed Integer Programming (MIP) formulation and non-linear constraints generation for SNOPT are presented in Section 4. This is followed by a discussion of how to implement simulated annealing (SA) and terminal propagation (TP) separately in Section 5 and Section 6. To test the performance of the methods, computational experiments were

carried out and the results are shown in Section 7. Finally, Section 8 concludes with a summary of this work and the conclusion is drawn therefrom.

3.2 Problem Description

In the floorplanning problem under discussion, blocks are assumed to be rectangular and have defined area. There are three kinds of blocks in this treatment. A *hard block* is a module whose width and height are given, but its position coordinates are arbitrary. A *soft block* is a module which can has various shapes as far as its aspect ratio is within a given range $[R_{u,min}, R_{u,max}]$. The position of soft blocks are also arbitrary. The last group of blocks called *pre-placed blocks* whose width, height and their coordinates are all fixed. The following notations are used to describe a block on the plane.

Parameters	Definitions
n	Number of blocks.
m	Number of nets in the netlist.
w_u, h_u	The width and height of block u.
x_u, y_u	The coordinates of the lower left corner of block u, referred to as the coordinates of block u.
w_k^b, h_k^b	The width and height of net bounding box of net k.
A_u	$A_u \equiv h_u \times w_u$, the area of block u.
R_u	$R_u \equiv h_u / w_u$, the aspect ratio of block u.
$R_{u,min}, R_{u,max}$	Lower and upper limits of the aspect ratio of block u.
w_c, h_c	The width and height of the chip's floorplan.

Since soft blocks have the most freedom, focus is restricted to the floorplanning

problem for soft blocks. The methods for hard and pre-placed blocks can be further extended based on the soft block analysis. The objective of the present floorplanning problem is to minimize chip size and total wire length. Chip size is defined as the total area of the smallest rectangle that encloses all blocks of the chip. While the total wire length is defined as the sum of the wire length for each of the netlists net. Here, a net is composed of set of pins, which belong to certain blocks and I/O pads inside this net. Because the positions of pins within soft blocks are unknown before exact shapes of the blocks are determined. It is assumed that the pins of each block are located at the block's center. To calculate the wire length of a net, *the half perimeter wire length* (HPWL) estimation method is used. This method provides a very good estimate for wire length approximation and is commonly used in other research [33], [39], [40] and [34]. The HPWL method consists of finding the smallest bounding rectangle (*net-bounding box*) that encloses all the centers of blocks in the net.

The following floorplanning problem is presented as a mixed integer nonlinear programming (MINLP) [34] formulation. The objective is to minimize:

$$\sum_{k=1}^{m}(w_k^b + h_k^b) \quad \text{and} \quad w_c * h_c.$$

3.3 Top-Down Methodology

This section discusses the overall floorplan design methodology. First a partitioning based method is used to break down the large input netlist into smaller sub-circuits. Secondly, a hierarchical approach is implemented such that at each step a small number of blocks (sub-circuits) undergo a finer floorplanning step. Therefore a top-down hierarchical algorithm structure is imposed. The partitioning based method employs *hMETIS* [37], while the finer floorplanning step is carried out with a non-

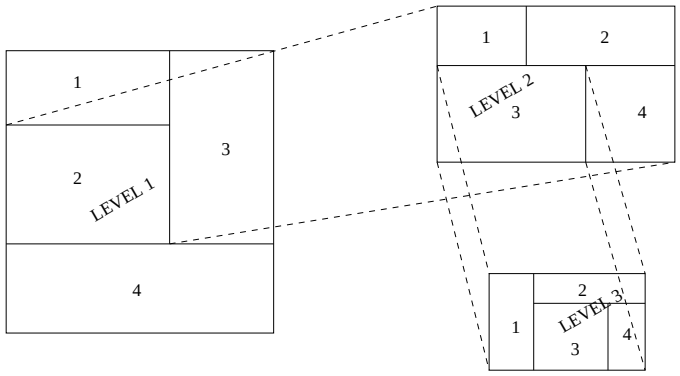

Figure 3.1: Hierarchical Floorplanning

linear solver package called *SNOPT* [25]. The software implementation developed in this work is called *Hercules* since it is capable of application to large industrial circuits as the *MCNC* [27] benchmark examples.

Hercules utilizes hierarchical floorplanning technique for large industrial circuits. A floorplan can be said to have a hierarchical structure if it can be achieved by recursively partitioning a rectangle into K (partition size) parts as illustrated in Fig. 3.1. The hypergraph partitioning package *hMETIS* [37] was used for this purpose. The input to *Hercules* was the *MCNC* benchmarks netlist and node areas in *GSRC* [27] format. The other adjustable parameters in *hMETIS* include the number of simulated annealing iterations and the partition size K. Additional details of the general framework and package implementation are discussed next.

The top-down algorithm begins with the partitioning of the input netlist into K parts using *hMETIS*, and is called the level one partitioning. After a series of preliminary tests, the *hMETIS* configurations are chosen such that vertices which are connected by the entire hyperedge are grouped together (CType=4). The load imbalance factor (UBfactor) is set to 5. The number of bisections (Nruns) performed is set to 10. The Fiduccia Mattheyses one-way scheme with refinement is performed. The refinement parameter (VCycle) is set to 3 which provides the best found solution

in a given number of iterations. Reconstruction is allowed, ensuring that the partitions retain the remaining portion of the hypergraph which were cut.

The K macros undergo the floorplanning step as explained in the previous section. The floorplanning step takes inputs like the bounding area (A), cutset coefficients and the areas (A_u) of the individual macros. For this stage the bounding area (chip area) is assumed to be the sum of the areas of all the nodes (blocks) with a certain area relaxation, restricted to $1\% - 5\%$. As far as the later nonlinear programming (SNOPT) can select a smaller percentage and place the blocks. Therefore, it becomes possible to achieve the objective of minimizing chip size. The individual areas ($wi*h_i$) of the macros are the sums of the areas of all the blocks within the macros. The bounding width (W) and height (H) at this level can be calculated as $\sqrt{A}$ since the chip is assumed to have a square shaped. The cutset coefficients α_{ij} used in the nonlinear model (SNOPT) are the number of nets that connect macro i and macro j after the partitioning process. After the first level floorplainning process, each of the soft macros width (w_i) and height (h_i) can be calculated using the same scheme ($w_i = h_i = \sqrt{A_i}$).

The second level partitioning step involves partitioning one of the K macros. To ensure that every group of nodes (blocks) is assigned a valid location on the chip. It is necessary to iterate through the node list and keep partitioning the macro which a target node belongs to. The recursive process stops when a point is reached for which the target node has no neighbors in the partition to which it belongs. Thus this target node can be placed with this particular partition's size and position. The algorithm is presented in pseudo code in the Fig. 3.2.

At each step the nodes in the macro under consideration are isolated. The sub-netlist is extracted from the original netlist information. The sub-netlist or the macro netlist includes only those nodes which are present in the macro. Then the partitioning is done by using *hMETIS* recursively. After the partitioning process, the parti-

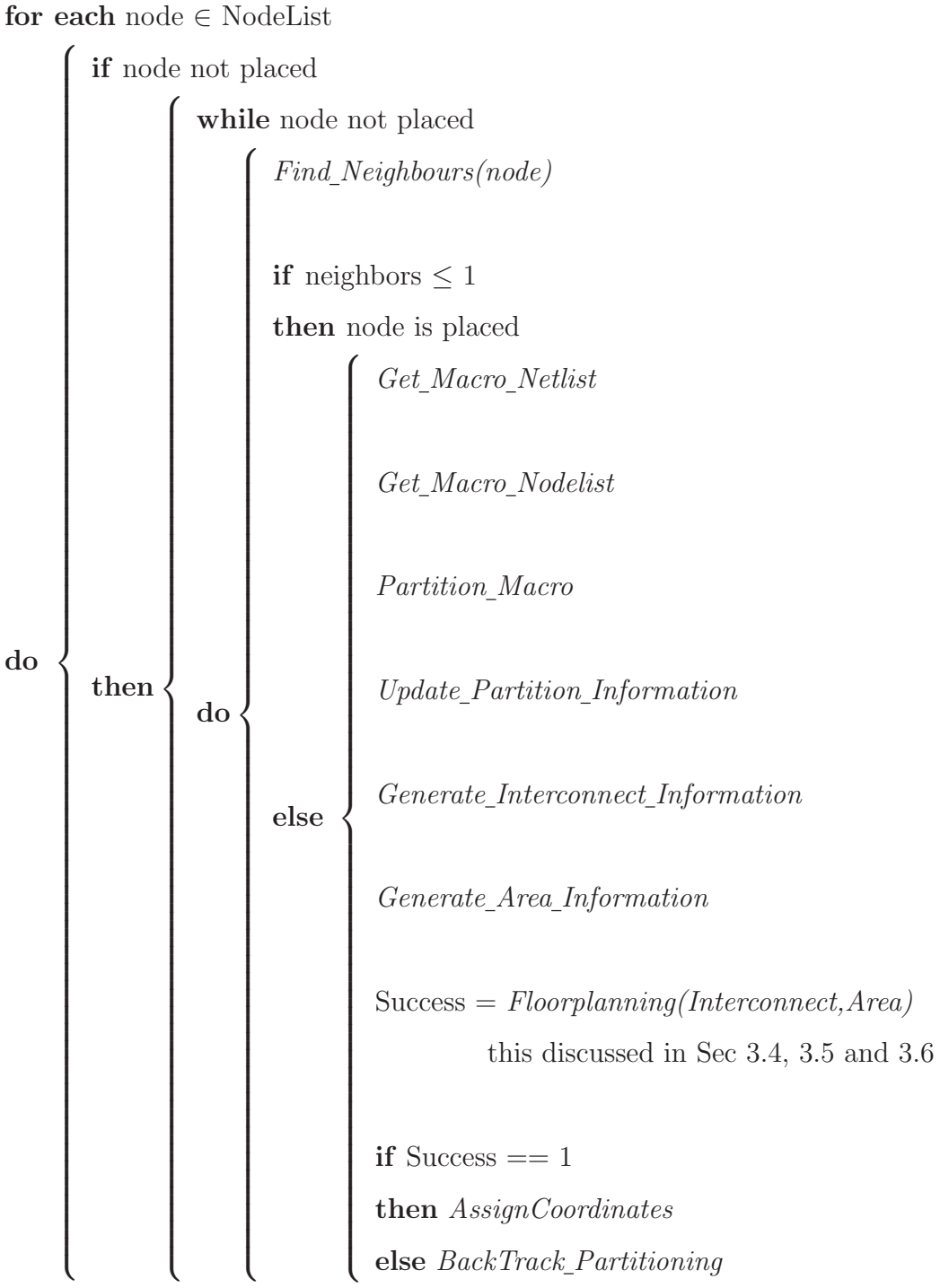

Figure 3.2: Top_Down Algorithm in Pseudo Code

tioning interconnect and area information are computed and are used to do the next *floorplanning(Interconnect, Area)* (Fig. 3.2). If for some reason the floorplanning step is unsuccessful then the changes made by the partitioning process is undone. When the algorithm encounters a node which belongs to a macro which could not be successfully floorplanned in the first attempt, the partitioning size K would be cut down for that macro before a second attempt is made. This makes the floorplanning problem easier by decreasing the number of variables.

3.4 Mixed Integer Programming Method

3.4.1 Mixed integer programming formulation

One of the most common ways to describe the floorplan topology is using sequence pairs [33]. However, here a slightly different method, called the Mixed Integer Programming (MIP) formulation using a pair of binary matrices, was used for this purpose [37]. It proposes an analytical approach using 0/1 integer variables to present the floorplan topology, and using integer programming to solve the floorplan optimization problem.

The main constraint in solving floorplanning area minimization (FAMP) [34] problems is the prevention of overlap. Considering the overlapping constraints of any pair of rectangular blocks i and j, there are four ways to position two macros such that they do not overlap each other. Let (w_i, h_i), (w_j, h_j) be the width and height of block i and j. Let (x_i, y_i) and (x_j, y_j) describe the x and y coordinates of the lower left corners of macros i and j. The overlapping constraints can be modeled as:

$$x_i + w_i \leq x_j \qquad \text{i is to the left of j} \tag{3.1}$$

$$x_i - w_j \geq x_j \qquad \text{i is to the right of j} \tag{3.2}$$

$$y_i + h_i \leq y_j \qquad \text{i is below j} \tag{3.3}$$

$$y_i - h_j \geq y_j \qquad \text{i is above j} \tag{3.4}$$

At this stage 0/1 integer variables $x_{i,j}$ and $y_{i,j}$ (Fig. 3.3) are added so that only one of the above inequalities is satisfied. In addition to the overlap constraints bounding constraints are also added to make sure that the macros are held within the bounding rectangle. If W and H are the width and height of the bounding rectangle then the bounding constraints are given by $|x_i + w_i| \leq W$ and $|y_i + h_i| \leq H$. W can be equal to sum of the width of all the macros while H can be equal to sum of heights of all the macros. Thus the equations can be rewritten as:

$$x_i + w_i - (x_j + W * (x_{ij} + y_{ij})) \leq 0 \tag{3.5}$$

$$y_i + h_i - (y_j + H * (1 + x_{ij} - y_{ij})) \leq 0 \tag{3.6}$$

$$x_i - w_j - (x_j - W * (1 - x_{ij} + y_{ij})) \geq 0 \tag{3.7}$$

$$y_i - h_j - (y_j - H * (2 - x_{ij} - y_{ij})) \geq 0 \tag{3.8}$$

First of all, these four overlap inequalities can be easily implemented in the non-linear model SNOPT's constraints. Secondly, from Fig. 3.3 it is seen that when handling the small number of macros in floorplanning, using 0/1 binary numbers is more clearer and easier than the sequence pair method [33] to represent the relative positions between two modules. And the top-down structure guarantees that only the small number of modules are floorplanned. Last but not least, when defining the

three different moves in simulated annealing step (SA) in Sec. 6, the MIP representation provides unique advantages when handling swap, move and rotation. For these reasons, the binary number expression in Mixed Integer Programming is preferred and not the commonly used sequence pair representation.

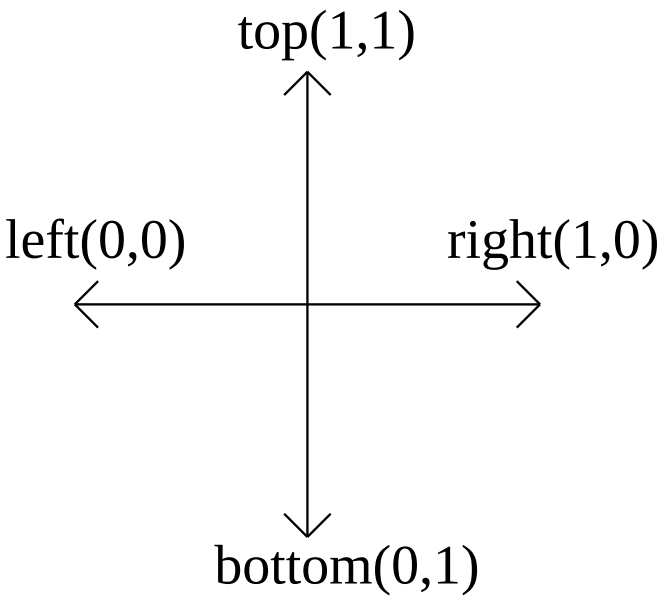

Figure 3.3: Mixed Integer Programming Formulaiton using binary numbers

3.4.2 Nonlinear model using SNOPT

Based on the previous discussion of the overall top-down design flow and the floorplan design representation form, it is appropriate to consider the nonlinear modelling of the floorplanning problem. This step corresponds to an important function $Floorplanning(Interconnect, Area)$, which is also mentioned in the top-down pseudo code in Fig. 3.2. The nonlinear solver used is called SNOPT [25], which employs SQP (Sequential Quadratic Programming) [52] methods to solve a MINLP (Mixed Integer Nonlinear problem) with smooth nonlinear objective functions, such as the floorplan

design problem under discussion. The modelling language used is AMPL, which gives the flexibility of hooking up any solver with the defined ampl model.

The macros considered for FAMP (Floorplan Area Minimization Problem) [34] are rectilinear in shape and are defined by width and height. The pin positions are assumed to be in the center and the Manhattan Half Perimeter Wire Length (HPWL) model is enforced. Since all the macros are considered to be soft blocks, which means they are only defined by their exact size and the width and height of each macro are not known. There is a need to specify the range of aspect ratio for each of the soft macros so that the final dimensions assigned are not too large or too small. The upper and lower ratio limit selected is $(R_{min}, R_{max}) = (0.1, 10)$. The bounding width W and the height H are given as the input parameters to the problem. The complete nonlinear model is given in Fig. 3.4.

The total bounding area considered is the sum of the areas of all the macros in the bounding box. No routing space has been allocated at this level and thus the total wasted area at this point is approximately zero. Therefore, the minimization objective in effect minimizes the total interconnect between the K (partition size) macros. The term l_{ij} specifies the manhattan wire length of the net that connects macros i and j. Each of these terms are multiplied by a cutset coefficient α_{ij}. The cutset coefficient defines the number of wires that connect the two macros which belongs in two different partition. This information is carried out from the previous hMETIS partition step. Thus the sum of the product of cutset coefficient and the manhattan wire length gives a measure of the total wire length inside the bounding box.

From (B.1) to (3.20) in Fig. 3.4, the terms x_{ij} and y_{ij} are the binary variables as introduced in Sec. 3.4.1, which are used to specify the referential position of each macro with the other. In other words a set of x_{ij} and y_{ij} represent the topology of floorplan. The overlap constraints prevent the macros from overlapping each other.

Minimize : L

$$L = \sum_{ij} \alpha_{ij} * l_{ij} \qquad \forall i, j \in S \tag{3.9}$$

$$x_{ij} = C \qquad C \in (0,1), \forall i, j \in S \tag{3.10}$$

$$y_{ij} = C \qquad C \in (0,1), \forall i, j \in S \tag{3.11}$$

Overlap Constraints :

$$x_i + w_i - (x_j + W * (x_{ij} + y_{ij})) \leq 0 \qquad \text{(i is to the left of j)} \tag{3.12}$$

$$x_i - w_j - (x_j - W * (1 - x_{ij} + y_{ij})) \geq 0 \qquad \text{(i is to the right of j)} \tag{3.13}$$

$$y_i + h_i - (y_j + H * (1 + x_{ij} - y_{ij})) \leq 0 \qquad \text{(i is to the bottom of j)} \tag{3.14}$$

$$y_i - h_j - (y_j - H * (2 - x_{ij} - y_{ij})) \geq 0 \qquad \text{(i is to the top of j)} \tag{3.15}$$

Boundary Constraints :

$$(x_i + w_i) - W \leq 0 \qquad \forall i \in S \tag{3.16}$$

$$(y_i + h_i) - H \leq 0 \qquad \forall i \in S \tag{3.17}$$

Area Constraint :

$$w_i * h_i = A_i \qquad \forall i \in S \tag{3.18}$$

Aspect ratio constrain :

$$R_{min} \leq \frac{w_i}{h_i} \leq R_{max} \tag{3.19}$$

Wire Length Constraints :

$$(x_i + w_i/2) - (x_j + w_j/2) + (y_j + h_j/2) - (y_i + h_i/2) - l_{ij} \leq 0 \tag{3.20}$$

$$(x_i + w_i/2) - (x_j + w_j/2) + (y_i + h_i/2) - (y_j + h_j/2) - l_{ij} \leq 0 \tag{3.21}$$

$$(x_j + w_j/2) - (x_i + w_i/2) + (y_j + h_j/2) - (y_i + h_i/2) - l_{ij} \leq 0 \tag{3.22}$$

$$(x_j + w_j/2) - (x_i + w_i/2) + (y_i + h_i/2) - (y_j + h_j/2) - l_{ij} \leq 0 \tag{3.23}$$

Figure 3.4: Non-linear model of the floorplanning problem

Each macro has an overlap constraint with every other macro. The binary variables using in MIP (Mixed Integer Programming) ensure that one and only one overlap equation is satisfied for each pair of macros. The bounding constraints are needed to ensure that each macro is confined to the bounding area. One of the most important constraints is the aspect ratio constraint. The aspect ratio of each macro is specified as a range and the lower and upper limits are defined as (R_{min}, R_{max}). The terms w_i and h_i are the unknowns in nonlinear program. However the area of each macro can be specified as the product of the w_i and h_i because of the use of a nonlinear modelling package SNOPT. This can be seen as a superior to other linear program methods. When using a linear program as [31] [33] [34], a linearized approximation can always be introduced, and this may further impact the effectiveness and optimization of the final floorplanning result. The final set of wire length constraints specify that the center-to-center manhattan distance is the minimum distance between a pair of macros. Only connected macros have wire length constraints.

3.5 Simulated Annealing (SA) Algorithm

In this study, we use an SA algorithm to find a combination of binary numbers which gives a valid relative position among all the macros and generates the best floorplan. In SA algorithm, an initial floorplan solution is repeatedly improved by making small changes until no further improvements can be made by such changes. Unlike greedy algorithms, SA algorithms can avoid getting trapped in the local optimum value by allowing occasional uphill moves which deteriorate the objective function. The uphill move is allowed with a probability given by $exp(-\triangle/T)$, where $\triangle$ is the difference between the current objective function values and neighborhood solutions, and T is a control parameter called the *temperature*. The temperature is initially assigned a large value which is gradually decreased by a predetermined method, called

the *Cooling Schedule*.

The following shows the three topology change moves and the main objective function of the SA algorithm.

3.5.1 Topology change moves

MIP ($MixedIntegerProgramming$) is used to represent two macro's position relationship as discussed in Section 3.4.1. By defining a pair of binary numbers (x_array, y_array), which is chosen from $(0,0), (0,1), (1,0)$ and $(1,1)$ as left, bottom, right and top shown in Fig. 3.3, the three basic moves can be easily implemented.

1. **Swap move** (Fig. 3.5)

   Two macros are randomly selected and only change the x_array in the corresponding pair of binary numbers, two macros swap their positions in horizontal or in vertical direction. The swap direction (horizontal/vertical) is decided by the remaining y_array (0/1).

2. **Rotate move** (Fig. 3.6)

   Two macros i and j are randomly selected and only change the $y_{ij}(y\_array)$ in the corresponding pair of binary numbers (x_{ij}, y_{ij}). if y_{ij} changes from 0 to 1, block j moves counter clockwise around block i. Otherwise, if y_{ij} changes from 1 to 0, block j moves clockwise. (with equal probability).

3. **Shift move** (Fig. 3.7)

   Two macros i and j are randomly selected, and if i and j are not neighbors (there are macros between them either horizontally or vertically), shift macro i and j together as neighbors. The choice of horizontal or vertical shift is by equal probability.

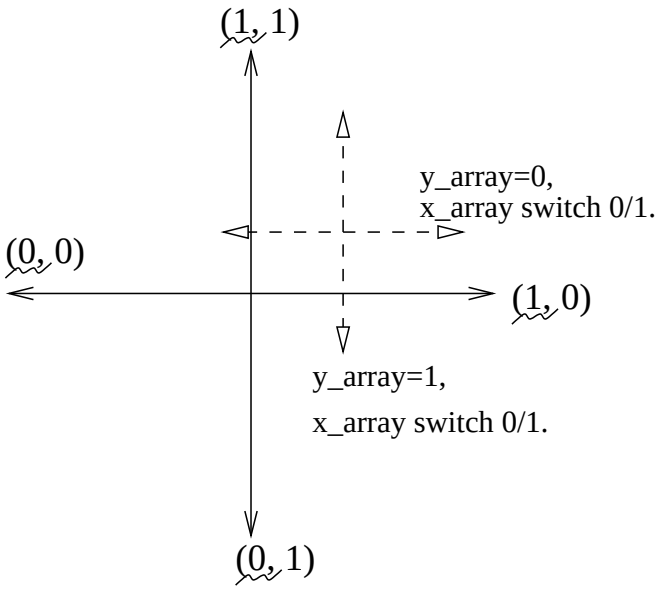

Figure 3.5: Simulated Annealing's Swap Move

In each iteration of the suggested SA algorithm, the three methods are randomly selected with probabilities 0.3, 0.3 and 0.4.

3.5.2 Objective function

Once a floorplan is generated from a topology change move, the total wire length is recalculated while considering both soft macro's pin position and chip input-output (I/O) pads.

$$WL_{total} = WL_{blocks} + WL_{pads} + WL_{TP} \tag{3.24}$$

From (3.24), the SA objective function composed of wire length for inter macros (L in Fig. 3.4), wire length between I/O pads and macros and wire length from terminal propagation (TP). The first two are both computed in half perimeter wire length ($HPWL$), the last one is an objective function adjustment because of the top-

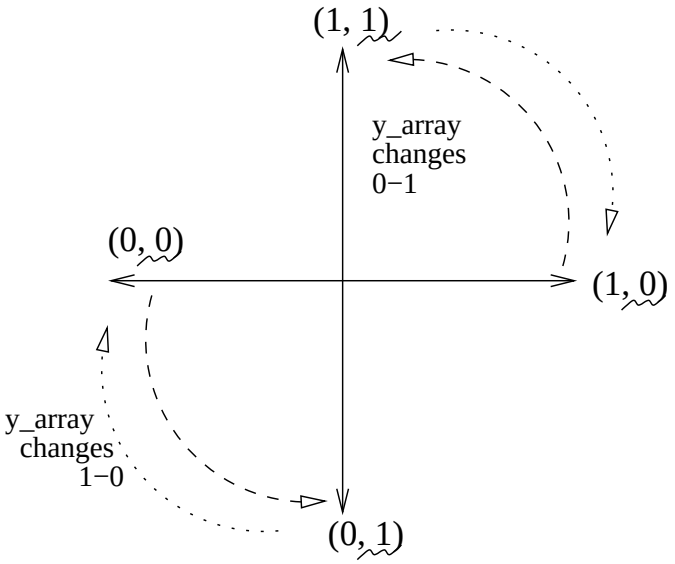

Figure 3.6: Simulated Annealing's Rotate Move

down hierarchical structure. The terminal propagation wire length is discussed in Sec. 3.6. Note that a floorplan generated by this method always satisfies all constraints of the problem because the above three topology moves are carefully selected. The SA algorithm can be applied only to valid topologies. The SA algorithm takes charge of reducing the total wire length at this level.

3.6 Terminal Propagation (TP) Algorithm

3.6.1 Reasons to import Terminal Propagation (TP) algorithm

Dunlop and Kernighan [53] first introduced the terminal propagation (TP) algorithm in the procedure for $VLSI$ circuits placement. They considered not only the internal nets of the subcircuit but also the nets connected to external modules at higher levels of the hierarchy. An example is shown in Fig. 3.8a, where the entire

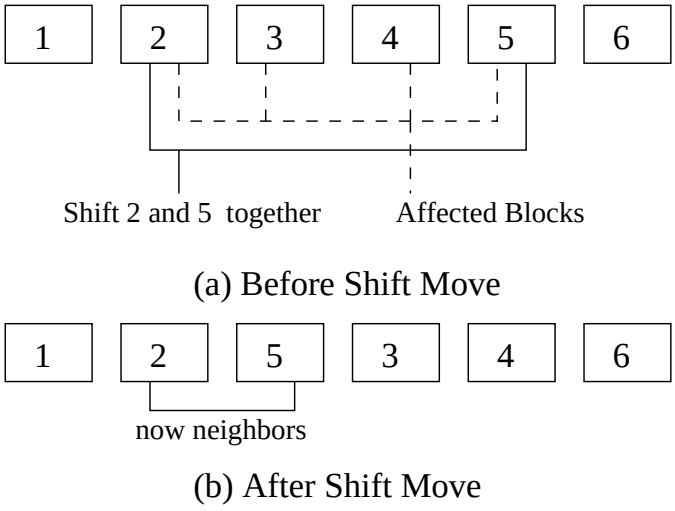

Figure 3.7: Simulated Annealing's Shift Move

circuit is divided into two sections. If a module is connected to an external terminal on the right side of the chip, the module should preferably be assigned to the right side of the chip. Thus the modules shown in Fig. 3.8a are in the right region. The method proposed by Dunlop and Kernighan is called the *terminalpropagation* algorithm, and is further explained in Fig. 3.8b,c. Consider the net connecting modules 1, 2 and 3 in block A. This net is connected to the other modules in blocks B and C. In Fig. 3.8c the modules in block C are represented by a "dummy" P2 on the boundary of block A. Similarly, the modules in block C are represented by a "dummy" module P1 on the boundary of block A. After partitioning, the net-cut would be minimized if modules 1, 2 and 3 were placed in the bottom half of A.

TP is the essential technique for the success of min cut type placement. It leads to better bisection results for placement compared to that of pure partitioning. This is because TP uses geometrical information of external terminals. Returning to consideration of the floorplanning problem, since the partition-based tool hMETIS was used to break down the large circuit, the circuit is separated into several levels. Be-

tween any upper and bottom level there are external connections just like min cut placement problem. It is necessary to import TP into Hercules top-down structure. In the following Sec. 3.6.2, the details about how to merge TP with Hercules are discussed.

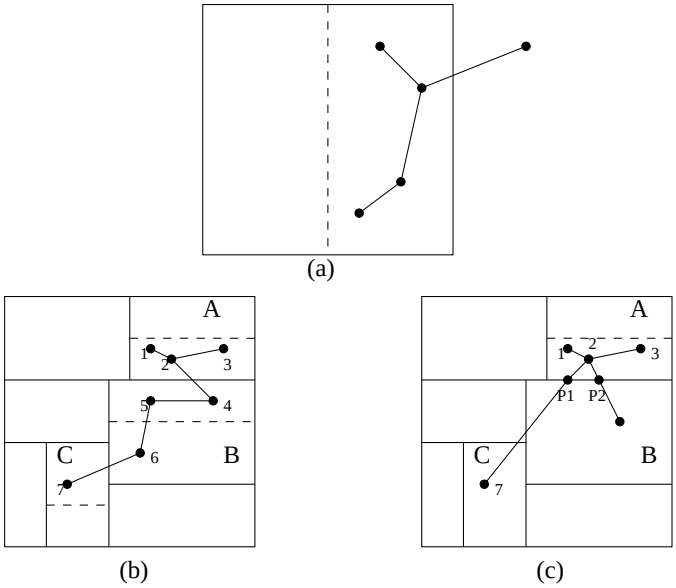

Figure 3.8: (a) Modules connected to an external terminal. (b) A net connecting modules in different blocks. (c) Modules in blocks B and C are replaced by dummy modules P1 and P2. Each point denotes a module or an external terminal.

3.6.2 How to implement TP with Hercules.

To add the TP algorithm to the original $Hercules$ top-down structure algorithm in Fig. 3.2, new core methodology pseudo code is introduced as shown in Fig. 3.9. After getting the partition information from $hMETIS$, the upper level dummy node connectivity information is collected and input to the nonlinear programming $SNOPT$ for calculating WL_{TP} (see equation 3.24). In order to calculate WL_{TP}, the fixed position around the specific block before nonlinear solver step must be assigned. Eight

fixed dummy points are introduced as shown in Fig. 3.10. All the upper level nodes beyond this block are mapped into these dummy points based on which region the extend nodes are in. For example, if a node A in region 2 has connections with the current level partitions, then A is mapped to point 2. Any connections between this extend node and the nodes inside the block will be mapped to the connections between the dummy point 2 and the sub-block that the inside nodes are in. To reduce the computation, only points from 1 to 4 in the programming are recorded, 5 to 8 are the exactly reverse points corresponding to these four points.

3.6.3 TP's experiments

The performance of the terminal propagation (TP) algorithm can be verified when doing floorplanning experiments in MCNC [27] testbench. For comparison reasons, these five testbenches were run using the two methods separately, one with terminal propagation, the other without terminal propagation. These two methods produced similar chip size and cpu time, but large percentage reduce when focusing on the final half perimeter wire length ($HPWL$). The simulate annealing iteration is 500. Each method was tested at least 10 times. And in each case, the optimum result was selected. The results are listed in Table 3.1. †a, b and c in (a, b, c) denote the numbers of soft blocks, nets and I/O pads, respectively. The reason why the apte-s testbench has no terminal propagation results is the small number of blocks, only 9 blocks exist. Therefore its floorplanning problem is treated in a flat level. The other four testbench HPWL results show that the terminal propagation algorithm optimizes the floorplanning problem effectively. The larger number of blocks that the testbench has, the more $HPWL$ decreases.

for each *node* $\in$ *nodelist*

while *node* not placed

P = the current level partition nodes

D = the upper level nodes which have connections with P,

(assigned to 8 points depict in Fig. 10)

IP = Interconnection in P

A = Area of partitions in P

ID = Interconnection between P and D

$success = Floorplanning(IP, A, ID)$

if *success*

update positions

else

BackTrack partitioning

Figure 3.9: Terminal Propagation with Hercules Top-Down Algorithm Core in Pseudo Code

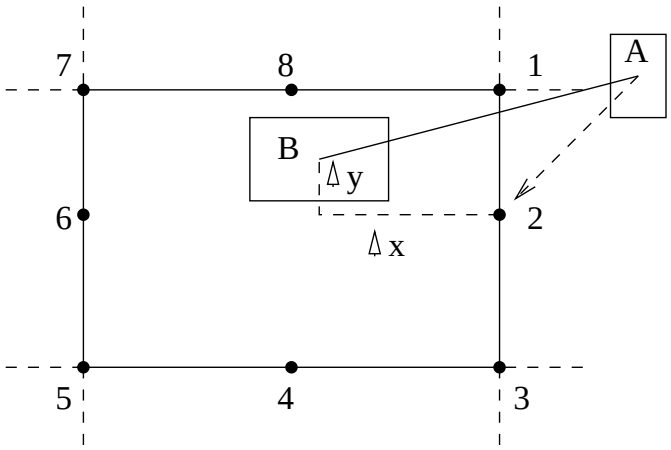

Figure 3.10: Dummy Points of a Partition

Table 3.1: Comparisons of Hercules Method with/without Terminal Propagation

	HPWL (μ)	apte-s (9,97,73)†	xerox-s (10,203,2)	hp-s (11,83,45)	ami33-s (33,123,42)	ami49-s (49,408,22)
Hercules	no TP	n/a	399201	128173	50526	748273
	with TP	303169	377217	120243	46107	655869
	percentage	n/a	-5.5%	-6.2%	-8.7%	-12.3%

3.7 Software Implementation

The Hercules program is written in C/C++ on a Sun-Blade Solaris platform. The nonlinear solver is fortran based and needs a Fortran based and therefore a Fortran compiler is required to run. The Hercules program can be invoked by the following command lines :

hercules <file in GSRC format> <nodes list> <Cluster Size> <Scale> <Simulation Iterations> <Partitioning Size> <TP parameter>

nets file :

The nets file is a GSRC format file which contains the circuit netlist information, including blocks and I/O pads connectivity information.

nodes list file :

The nodes file is a GSRC format file which contains all the blocks area, and I/O pads pre-fixed positions.

Cluster Size :

This parameter is designed to set the clustering or partitioning approach. If the parameter is greater than 0 then it defines the max clustering size allowed. This feature is still under construction and should be disabled by setting the parameter to 0.

Scale :

This parameter is used to scale the areas of the blocks. This was added because the solver cannot handle large values and it may be necessary to scale the areas initially. However the total wire length is computed in a way that takes the scaling factor into consideration.

Simulation Iterations:

This parameter sets the partitioning size **K** to be used at each level of partitioning.

TP Parameter:

Is the switch parameter which determines whether or not the user wants to implement terminal propagation into Hercules.

After a successful run the program will generate three files. The .place file contains the coordinates of each block with respect to the chip boundary. This file also contains the time required for execution, the levels of partitioning, the number of times hMETIS has been called, the Kruskal's minimal spanning tree (MST) [44] [45] wire length and the conventional half perimeter wire length (HPWL). The .netlen file describes the amount contributed by each net to total wire length. The .mag file can be used to view the floorplan results graphically by using Magic [46] layout software.

3.8 Design Limitations

The top down design approach has a few limitations as does the approach used in this work. One of the most important issues is global optimization of the problem. The approach used involved breaking up a complex problem into a number of simpler problems and using the solutions of these sub-problems to get the solution to the complex problem. Thus many locally optimized solutions are computed in the hope that these will eventually lead to the final globally optimized solution. This however may not be the case. In the context of the approach employed a group of four to eight macros undergo floorplanning and the interconnect optimization is done between these groups of macros. Thus when a partition undergoes the floorplanning step the partition gets isolated since the neighboring interconnect information is not used. The macros having connections external to the partitions they belong to may move further away from each other and increase the total wire length. The partitioning process focuses on reducing the cut set so that the partitions are not highly connected. But even an initially insignificant wire at level one could become critical at a later high level since the length of the wire eventually increases if the cells it connects move away

from each other. The situation can be explained as the assumed non-critical wires become of more and more critical importance as the top down process continues. This may also lead to timing delays and reduce the speed of the circuit or in the worst scenario the circuit may fail to work. A possible solution to this problem is discussed in the future work section.

3.9 Computational Experiments

In this study, the Hercules top-down method for generating a floorplan from mixed integer programming (MIP) was compared with the algorithm of Murata and Kuh [33] and Kim *et al.* [34] and on the Microelectronics Center of North Carolina (MCNC) benchmark examples (apte-s, xerox-s, hp-s, ami33-s and ami49-s). For the recursive partition step using $hMETIS$, the unified partition size was fixed at 4. For the simulation annealing algorithm suggested in this study, the iteration number was set to 500. The experiments were done on a 500-MHz SUN Blade work station and $SNOPT$6.0 was used to solve nonlinear programs with the Hercules algorithm.

The two algorithms of Murata and Kun [33], are known as EXACT and DIS2+POST, and they were obtained from a 250-MHz DEC Alpha work station. The Kim *et al.* [34]'s algorithm named SA-CT+LP and SA-LP, were used to test the benchmark on a 500-MHz Pentium III processor. Both of Murata and Kim's methods use the sequence-pair to represent the topology of a floorplan and use a SA algorithm to find the best sequence-pair in a single flat level. In the test problems, both the lower and upper limits on the aspect ratio of the chip were set to 1.0, thus the chip should be of a square shape (Fig. A.1 – Fig. A.5).

Table 3.2 shows the detailed results of the comparison. First, compared with EXACT and DIS2+POST from Murata *et al.*, the floorplans obtained from the algorithms suggested in Hercules are much better in terms of the half perimeter wire

length (HPWL) and slightly better in terms of the chips size except for hp-s. Secondly, if SA-LP, which has the best performance over current algorithms, is used, Hercules gives very close values ($< 1\%$) of Area and HPWL in three out of five benchmarks (xerox, ami33 and ami49) with a much shorter computation time. Particularly for ami49 the largest benchmark case, Hercules shows slightly better performance for Area (-0.18%) and HPWL (-1.2%) metrics with the computation time almost 250 times faster than SA-LP. The best final floorplanning graphs for the MCNC benchmarks are also provided in Appendix (Fig. A.1–Fig. A.5).

3.10 Conclusion

In this chapter, the floorplanning problem has been analyzed in some detail to obtain the objective function of minimizing the total wire length and the chip size. A new top-down hierarchical floorplanning structure is given for the problem, and mixed integer programming formulation is used to represent the topology of floorplans. Inside the top-down frame, hMETIS is first implemented as a partition tool to break down the original large circuit into different levels. Within a partition level, the finer floorplanning problem was solved by a nonlinear solver called $SNOPT$. The overall wire length objective minimization ($WL_{tot} = WL_{blocks} + WL_{pads} + WL_{TP}$) was embedded in SA algorithms, which are used to find a near-optimal floorplan. After taking into account the external connections between different partition levels, the terminal propagation method was used and the improvement was quite effective ($HPWL$ average decreased 8.2%, and the largest benchmark ami49 $HPWL$ decreased as much as 12%). Results of computational tests on well-known benchmark problems confirmed that $Hercules$ gives good or similar solutions compared with the existing algorithms, and the respective cpu times are shorter especially when handling the large number of blocks benchmark examples.

Table 3.2: Comparisons of Hercules Method with Murata *et al.*'s EXACT, DIS2+POST and Kim *et al.*'s SA-CT+LP, SA-LP

		apte-s (9,97,73)†	xerox-s (10,203,2)	hp-s (11,83,45)	ami33-s (33,123,42)	ami49-s (49,408,22)
EXACT	Area(μ^2)	46553329	19509889	8826841	1159929	35581225
	HPWL(μ)	344358	401254	118819	53393	775104
	cpu (sec)	789	1198	1346	75684	612103
DIS2+POST	Area(μ^2)	46635241	19474569	8868484	1162084	36048016
	HPWL(μ)	421174	533990	157166	51346	850305
	cpu (sec)	1.32	2.06	2.05	28.09	50.40
SA-CT+LP	Area(μ^2)	46726924	19352812	8835280	1158363	35613678
	HPWL(μ)	243412	452686	121519	51383	746924
	cpu (sec)	2.74	3.92	5.01	115.1	402.4
SA-LP	Area(μ^2)	46561640	19350308	8830586	1157829	35520644
	HPWL(μ)	278178	376092	101011	45759	663689
	cpu (sec)	496.5	4872.0	1659.9	16260.0	68995.1
Hercules	Area(μ^2)	46588820	19359094	8836351	1158809	35457336
	HPWL(μ)	303169	377217	120243	46107	655869
	cpu (sec)	48	50	61	190	277
	HPWL→SA-LP	+8.9%	+0.3%	+19%	+0.76%	-1.2%
	Area→SA-LP	+0.058%	+0.045%	+0.065%	+0.085%	-0.18%
	cpu→SA-LP	$\frac{1}{10.3}$	$\frac{1}{97.4}$	$\frac{1}{27.2}$	$\frac{1}{85.5}$	$\frac{1}{249.1}$

However, there are some potential future work to do. Since Hercules uses hMETIS to first partition the original large circuit into smaller one. The partition size K is a very important parameter which has non-replaceable influence throughout the floorplanning design. If the inner structure characteristics in a circuit netlist [54] can be found before the floorplanning design, and this information used to decide every partition level's partition size K, the final solutions will be further improved. Also, the nonlinear solver SNOPT has limitations when handling large number of variables. Reducing and merging different variables and constraints is worthy of additional study.

Chapter 4

ILP Based Task Assignment and Scheduling for Collaborating Unmanned Autonomous Vehicles

4.1 Introduction

Unmanned Autonomous Vehicles (UAVs) are emerging as a breakthrough technology for numerous applications in manufacturing, military, environment monitoring, infrastructure monitoring, and so on [55], [56], [57]. These applications involve a functionality (behavior) that must be achieved under strict constraints (e.g., deadlines, precision, safety, energy resources, etc.), and optimized costs, such as the value (utility) of the achieved objectives, the time required to meet the objectives, and the spent energy. In addition, UAVs must safely operate in hard-to-predict or even unknown environments and conditions, including moving obstacles and dynamically emerging threats [58], [59]. This poses interesting new challenges for the development of decision making (control) systems, which ought to provide optimized and reliable response in both predictable and hard-to-predict instances.

UAVs sense the environment through a rich set of sensors and compute a response that is delivered through various actuators, including those used for moving. UAVs perform a large set of activities, including the computations of trajectories and iden-

tification of the control values for travelling along a trajectory, signal sampling and processing (including image processing), communication with other UAVs as well as specific activities, such as target detection, handling and assessment [60], [61], [62], [57].

In many situations, UAVs must often operate collaboratively, so that complex activities can be undertaken jointly by a group of UAVs [55], [56], [57], [63], [64]. The nature of collaboration is often decided dynamically at run time, depending on the context-specific situations. For example, a UAV might not be able to meet the deadlines set for the tasks due to unforeseen overheads, such as the time required to avoid moving obstacles. In this case, the UAV interrogates other neighboring UAVs, some of which may have more flexible deadlines, if they can participate in a collaborative effort.

A major challenge for UAVs is the requirement that they are able to provide predictable and reliable operation in hard-to-predict environments and situations. Traditionally, reactive control has been the *de facto* solution for functioning in situations that cannot be characterized off-line. Depending on the conditions which are identified on-line, the controller selects the most suitable response from a set of predefined strategies. Each response strategy is characterized by specific outcomes and performance, such as execution time, energy consumption, and so on. While certain "fixed-point" properties can be proven for reactive behavior (like stability and reachability) [63], [65], [66], properties which depend on dynamic attributes (e.g., the frequency of being in a state) are harder to prove unless restrictive functioning conditions are assumed. Thus, important performance metrics, e.g., execution time and resource (energy) consumption, are hard to accurately estimate and guarantee for reactive control procedures. The alternative to reactive procedures are off-line static control methods [56], [67]. These methods work very well if the operational conditions and the environment are fully known, and hence the UAV behavior is

deterministic. The performance of the control methods can be precisely estimated in this case. However, static methods have little or no flexibility in adapting to unknown situations. Therefore, it is challenging to devise general control strategies with predictable performance and that operate efficiently in hard-to-predict conditions.

This work is based on an approach to devising performance predictive methods for collaborative control of UAVs operating in environments with fixed and unknown (pop-up) targets. The control strategy allocates targets to specific UAVs and schedules in time the handling of the targets assigned a particular UAV, so that the deadline associated with the fixed targets are all met. In addition, the method maximizes the chances of pop-up targets being properly handled by UAVs. The behavior of pop-up targets is described in terms of probabilistic distributions for their appearance in a given zone and time interval. The third optimization criterion is to maximize the flexibility of UAVs to participate in collaborative actions in which multiple UAVs participate jointly to handle the same fixed target. This goal is addressed by computing the slack time margins that can be used by a UAV in collaborative actions while still meeting the deadlines of its assigned tasks. Please note that the approach does not select the UAVs statically that participate in collaboration since this would limit their operational flexibility in unknown situations. Instead, each UAV decides on-line whether or not it will respond to a request for collaboration depending on its then available slack time. If the slack time is less than the computed margin then the UAV can not participate without violating the deadlines.

The proposed decision making approach assumes a two-level control hierarchy: the upper level strategy is a static procedure which determines the allocation and scheduling of the fixed targets to individual UAVs. The lower level strategy is reactive, and controls the handling of pop-up targets and a UAV's responses to requests for collaborative activities. The reactive components decisions are based on inputs from the UAV's sensors as well as the slack time constraints allocated to the UAV through

the static decision making process. An Integer Linear Programming (ILP) based solution for assigning and scheduling the fixed targets to UAVs and computing the slack time intervals used for collaborative actions and pop-up target handling is the basis for the work that follows.

This chapter is organized into six sections. Section Two summarizes related work. Section Three defines the addressed problem, and presents the modeling solution used for assigning and scheduling tasks to UAVs. Section Four discusses the ILP modeling of the problem, and a case study is described in Section Five. Finally, conclusions are presented.

4.2 Related Work

This section summarizes existing work on control for autonomous systems.

Brogan and Hodgins [68] present distributed control techniques for groups with significant dynamics. Similar to the method by Reynolds, there are only simple interactions between neighboring individuals. Interactions include communication, cooperation, and coordination strategies. Individuals have inertia, which imposes restrictions on the position and velocity gradients. Therefore, the computed trajectories and velocity control rules must not only address collision avoidance and flocking but also prevent steep changes in direction and velocity.

Brock and Khatib [69] describe a method in which distributed control is achieved through superposition of global and local requirements. Global requirements are static (e.g., the starting and ending points of a trajectory), and are modeled similar to virtual elastic strips that would span the tow points. Any changes from the trajectory, such as to avoid an obstacle, introduces an elastic force that attempts to pull back the robot to the planned trajectory. Local requirements, such obstacle avoidance, are expressed through local interactions, i.e. repulsive forces between obstacles and

moving robots. Then, the elastic and repulsive forces are used for velocity tuning and trajectory modification. In addition, the method relies on global information, e.g., robot priorities, and global trajectory re-planning to avoid any collisions. The method by Yamaguchi [64] uses attraction forces to targets and neighbors, and repulsive forces to obstacles for coordinated pattern formation for security against invaders. The author proves stability of the control rule, but does not suggest a general method for finding the local behavior rules of global requirements.

Leonard and Fiorelli [70] suggest a 2D distributed vehicle control based on artificial potentials and virtual leaders. Vehicles interact following potential fields following a logarithmic law. There is an attraction force toward distant neighbors, so that group forming is encouraged, and there are repulsive forces and velocity matching with the close neighbors. In addition, virtual leaders define an additional local potential, which aids group formation. Finally, dissipative forces attempt to match a vehicles velocity to a desired velocity value. Vehicle control is guided by the forces defined by logarithmic laws for the four potential fields. The authors also describe the mathematical rules that lead to certain formations, such as an equilateral triangle and hexagonal lattice.

Another artificial field-based approach is described by Mamei, Zambonelli, and Leonardi [71]. Distributed control is based on computational fields (called Co-Fields), which act as an enabling global infrastructure between mobile agents. Co-fields are dynamic, and are influenced by moving actors. Actors move along the field waveforms, following either downhill, uphill, or equipotential lines. The goal of vehicle control is avoiding collisions, forming predefined patterns, or vehicle meeting, but the authors suggest that finding the Co-Fields required for a certain goal is still an open research problem. The method presented in [59] combines nonlinear model predictive control with potential field concepts for conflict-free trajectory planning. Other field-based control methods are discussed in [72], [65].

Schouwenaars, Valenti, Feron, How, and Roche [67] discuss UAV trajectory generation for cooperative missions between manned and unmanned vehicles. A MILP (mixed integer linear programming) method is proposed for optimized real-time trajectory planning to avoid obstacles and threats in a partially-known environment. A cooperative task scheduling method is presented in [55].

Rathbun *et al.* [58] propose path planning algorithms for UAV navigation in uncertain environments. The position of obstructions is known only within a limited accuracy. Trajectories are found using evolutionary algorithms (EA) that implement the following operators, (i) mutate and propagate, (ii) crossover, (iii) goto goal, and (iv) mutate and match. Obstacle intersection probability is defined for both moving and fixed obstacles. As optimization is carried out, more information is gathered about the position of the obstacles, which improves the accuracy of the probability functions. The cost function relates the goal to reach the desired target, minimizing the amount of used fuel, and minimizing the probability of UAV collision with an obstacle. As the position of the obstacles becomes known, trajectories are replanned such that they maximize the probability of success, and minimize the change in trajectory (as compared to the already existing trajectory). Other EA based approaches are presented in [62], [61].

Trajectory planning based on graph search methods using A* and D* methods are discussed in [73], [74].

4.3 Problem Description and Modeling

This section defines the addressed problem and presents the proposed modeling.

Figure 4.1 summarizes the UAV behavior scenario. Multiple UAVs must cooperatively handle targets located in a 3D environment. Each UAV moves along a non-linear trajectory at a variable speed using a trajectory computing algorithm sim-

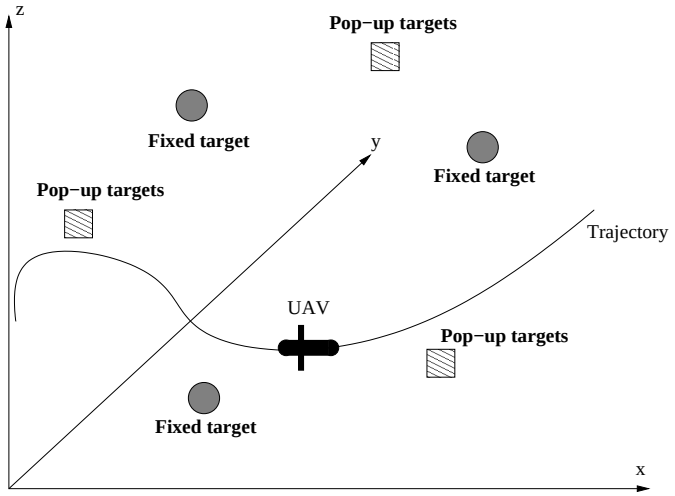

Figure 4.1: UAV movement towards fixed and pop-up targets

ilar to [75], [76]. UAVs are heterogeneous, and therefore may have different dynamic characteristics (e.g., speed and acceleration). The speed magnitude and speed gradients are bounded.

Targets are of two types: fixed targets and pop-up targets. Fixed targets are positioned at known locations, and must be handled before a predefined time limit. Otherwise, the entire mission is considered compromised. Pop-up targets are positioned at random places within a given 3D region, appear at random time instances, and disappear after a random time interval. The group of UAVs must handle all fixed targets before their time limit expires, and also maximize the probability of handling as many pop-up targets as possible. Fixed obstacles are present in the 3D environment, and must be avoided by the moving UAVs.

Handling targets consists of the following sequence of activities: (i) flying to the target location, (ii) detecting the target (e.g., through different sensors), (iii) completing certain target-related tasks such as taking pictures of the target, and (iv) assessing the results of the described activities [56], [67]. The latter three activities have known execution times, but the first step may be of variable duration, depending

on the position and flight characteristics of the UAV. It is assumed that each UAV can handle multiple fixed and pop-up targets, and that UAVs can collaboratively handle the same fixed target. For example, one UAV might detect and handle the target, and then transmit all the information to another UAV, which will do the assessment of the results for that target. This scenario is useful considering that UAVs have different capabilities (e.g., achievable speed), and resources (such as amounts of fuel). A slower UAV positioned nearby can do the assessment, while the more powerful UAV moves towards handling the next target. As a trade-off, the collaborative scenario involves communication overhead for information transmission between the UAVs, and also additional distances to be travelled (thus, higher fuel consumption) by the UAVs involved in collaboration.

Pop-up targets can be handled while a UAV flies to the target. If a pop-up target appears, the UAV might decide to carry out the necessary tasks related to that target, if the time required to do so does not result in violating the timing constraints for that UAV with respect to its assigned fixed targets. If time permits, multiple pop-up targets can be handled before the UAV arrives at the next fixed target.

In summary, this section has discussed the following problem: An algorithm must be developed for handling N fixed targets by M UAVs with known but different characteristics. Each fixed target must be handled before its predefined time limit expires. In addition, the probability of handling as many pop-up targets as possible must be maximized.

4.3.1 Collaborative approach

Figure 4.2 presents the decision making approach proposed for controlling the behavior of UAV groups. The approach represents a trade-off between centralized decision making, which is efficient and offers predictable performance (e.g., guaran-

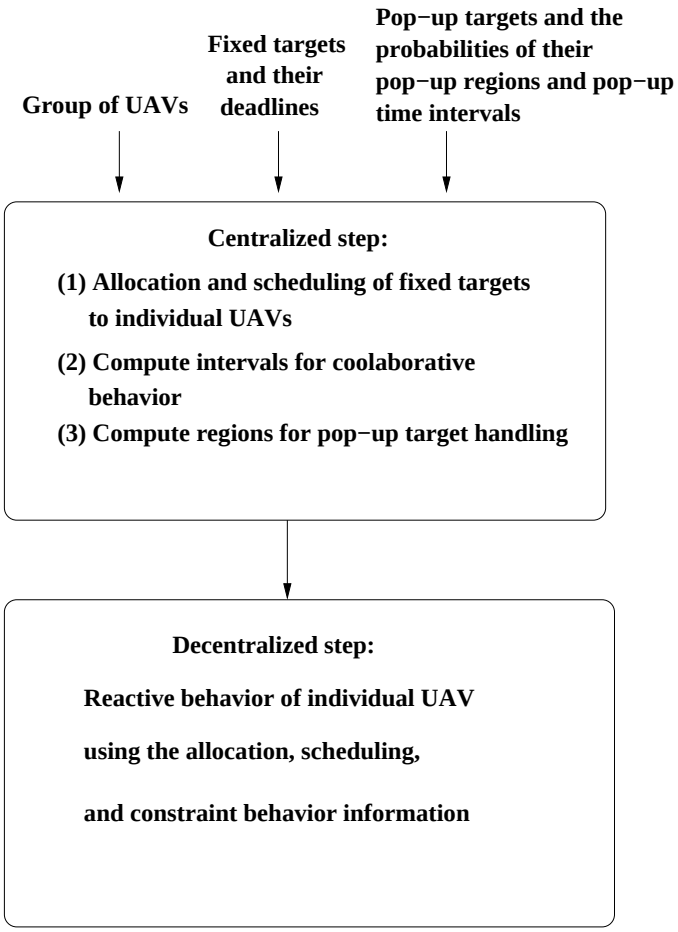

Figure 4.2: Decision making for collaborative UAV operation

tees the satisfying of the predefined deadlines), and decentralized control, which is more scalable and more flexible with respect to handling new situations. Centralized decision making is at the level of the entire UAV group, and decentralized control is at the level of each individual UAV.

This approach uses an off-line, centralized decision making step to compute the allocation of fixed targets for each UAV, and the temporal scheduling of the activities related to the handling of a target. In addition, the centralized step also calculates the constraints that encompass the collaborative behavior of each UAV. During operation, each UAV decides dynamically (after a collaboration request has been received) whether it will participate to the collaboration, or not. The decision is made based on its current situation, such that the deadlines of the allocated fixed targets are not violated. Since collaboration requests are formulated dynamically and cannot be predicted a priori, the optimization goal is to maximize the chances of a UAV participating in collaborations by computing the allocation and scheduling solution that maximizes its flexibility to participate in collaborations.

Similarly, the centralized decision process calculates constraints that maximize the likelihood of handling pop-up threats while satisfying the deadlines of the fixed targets. The handling of pop-up threats is also decided locally by each UAV during operation depending on its current situation and the deadlines of the assigned targets.

Each UAV executes its own decentralized controller, which implements a reactive behavior expressed through a Finite State Machine (FSM). The controller determines the specific actions of a UAV, e.g., pursuing a fixed target, satisfying a request for collaborations, or handling a pop-up threat. Figure 4.3 shows the structure of the reactive controller. In normal operation mode, the UAV is handling a fixed target following the allocation and scheduling decisions of the centralized step. If the controller detects that the deadlines fixed for the target currently being handled cannot be met, it formulates a request for collaboration. If the request is granted by another UAV

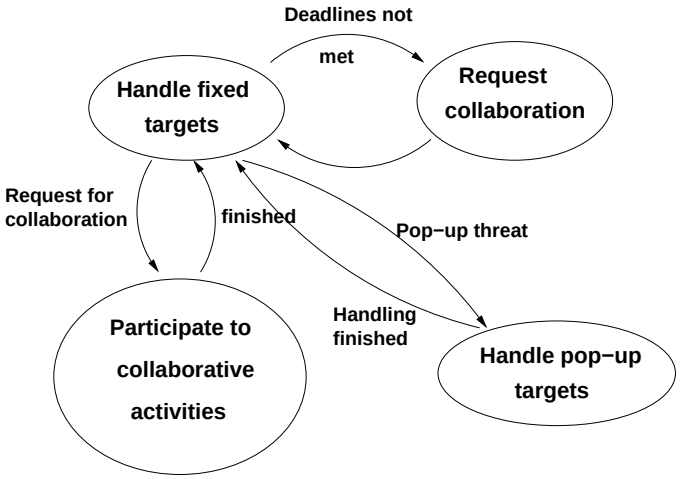

Figure 4.3: Decentralized controller for UAV operation

then the UAV moves on to handling the next assigned fixed target. If the request is not granted then the UAV might decide to continue with the current activity even though this results in violating the target's deadline, or leaving the task unfinished in order to move on to the next assigned target. Similarly, pop-up targets are handled if the related constraints allow it.

The focus of this discussion is on the centralized decision making algorithm, including target allocation and scheduling, as well as the computing of constraints related to the collaborative actions and pop-up threat handling required for the decentralized controller. The next section presents the proposed UAV behavior modeling used for centralized decision making.

4.3.2 Problem modeling

Figure 4.4 illustrates the task graph for handling N fixed targets. Each target handling activity is an independent thread of tasks consisting of separate tasks for flying, detection, handling, and assignment. Task dependencies express the required order of performing the tasks. Since the N targets are known, the times for detection,

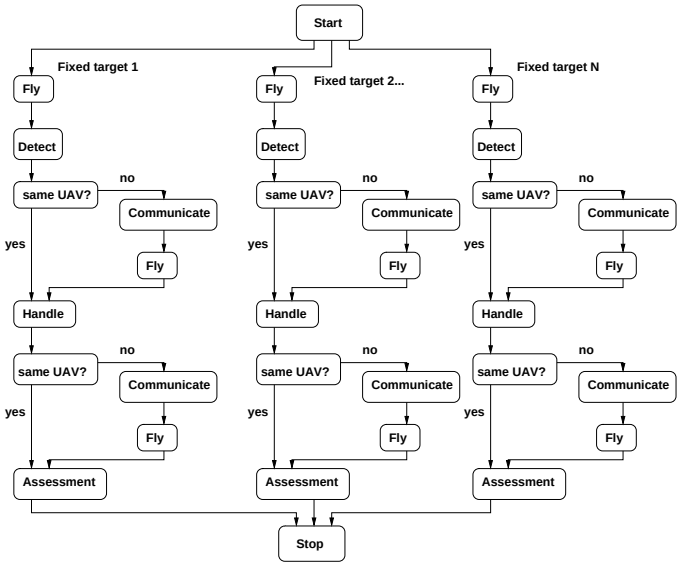

Figure 4.4: Task graphs for fixed target handling

handling, and assignment are fixed for a given UAV type. Note that these times are different for UAVs with different characteristics. In contrast, the flight time is not known in advance because the time required for reaching a target depends on the computed UAV trajectory and the decision to handle any encountered pop-up target. Moreover, the UAV trajectory depends on the position of the fixed target previously handled by the UAV, and therefore on the previous decisions on target allocation and scheduling.

1. **Fixed targets**

   The task graph includes tasks that can represent a collaborative behavior between multiple UAVs in handling the same fixed target. For example, after the target was detected by an UAV, the methodology allows other UAVs to handle and/or assess the target. These actions are represented by conditional blocks (the blocks labelled as "same UAV?" in the figure), which continue with the tasks for communication and flight, if a different UAV is involved. The com-

munication time is fixed (since the amount of data to be transferred is given), but the flight time is variable as it varies with the position of the UAV entering the collaboration. The flight time includes the total time spent by a UAV for moving for a new activity as well as the time for accomplishing that activity. Since the nature of the collaborative activity is decided online, the actual flight time is not known during the off-line centralized decision making step, and instead the methodology should maximize the overall capability of a UAV group for collaborative activity.

2. **Pop-up targets**

   Figure 4.5 summarizes the main characteristics of pop-up target modeling. Pop-up targets can randomly appear at any point in a known regions. For example, in the figure, the pop-up target can be positioned at any point in the cubic volume. The time at which a pop-up target appears, and the length of time that the target is observable are random. This discussion assumes that the probability of a pop-up target appearance is constant over a given 3D volume and also constant for a given time interval. However, this methodology also supports more sophisticated scenarios in which probabilities follow distributions that vary over time.

4.4 Proposed Algorithm

This section describes the centralized task assignment and scheduling problem as an Integer Linear Programming (ILP) problem. The model is then solved using an existing ILP solver to obtain the centralized controller of a UAV group. Figure 4.4 is provided to explain the ILP expressions. The following equations are used to build the ILP model:

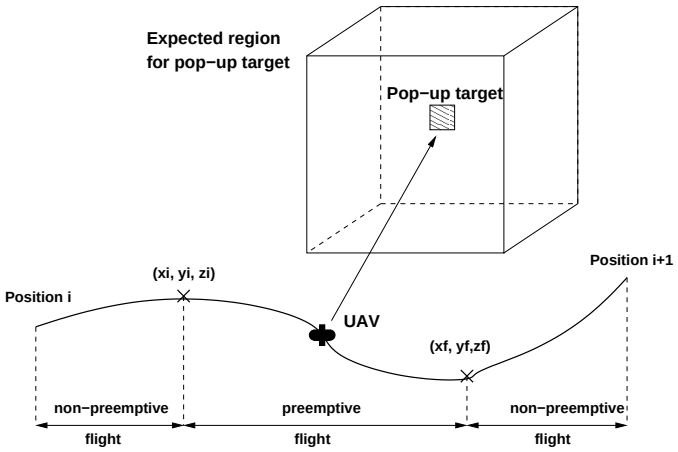

Figure 4.5: Strategy for pop-up targets

4.4.1 Task start time

The start time of a task i is larger than the end time of its preceding task j:

$$t_{i,start} \quad \geq \quad t_{j,end} \tag{4.1}$$

4.4.2 Task end time

$$t_{i,end} = t_{i,start} + x_{1,i}\,T_1 + x_{2,i}\,T_2 + ... + x_{M,i}\,T_M \tag{4.2}$$

The end time for executing a task (e.g., detection, handling, and assessment) is equal to the start time of the task plus the time T_i required for UAV_i $(i = 1, ..., M)$ to perform the task. Values of T_i are constants for a set of UAVs. The variable x_i is one, if the task is performed by UAV_i, otherwise it is zero.

4.4.3 Task allocation to UAVs

Each task pertaining to a fixed target must be allocated to the specific UAV, that performs the task. For task k, this requirement is expressed as follows:

$$\sum_{i \in UAVs} x_{i,k} = 1 \tag{4.3}$$

4.4.4 Task scheduling to UAVs

Each UAV can handle multiple fixed targets, one target at a time. The set of ILP equations must include relationships that constraint the UAV to execute a single task at a time. For the tasks pertaining to the same fixed target, these constraints are implicitly introduced by the equations 4.1, which represent the sequencing constraints of the tasks.

For the tasks related to different fixed targets allocated to the same UAV, the constraint is that the UAV handles a new target only after it finished handling the current target. Allowing the UAV to intertwine the handling of the two targets would result in unnecessary temporal overhead due to the extra distance the UAV must travel between the two fixed targets. This overhead obviously affects the optimality of the scheduling result.

A 0/1 variable $z_{i,j}$ is defined for each pair of fixed targets i and j. If both targets are handled by the same UAV, then the variable being one indicates that task i is handled before task j, and after task j, if the variable is zero. This constraint is captured as follows:

$$T_{i,end} \leq T_{j,start} z_{i,j} \sum_{k \in UAVs} x_{k,i} x_{k,j} + T_{\infty}(2 - z_{i,j} - \sum_{k \in UAVs} x_{k,i} x_{k,j}) \tag{4.4}$$

$$T_{j,end} \leq T_{i,start}(1 - z_{i,j}) \sum_{k \in UAVs} x_{k,i} x_{k,j} + T_{\infty}(1 + z_{i,j} - \sum_{k \in UAVs} x_{k,i} x_{k,j}) \tag{4.5}$$

T_{∞} is a very large value.

4.4.5 UAV flight time to fixed targets

The flight time T_{fly} to a fixed target depends on the previous position of an UAV, which results from the fixed target allocation and scheduling. Target allocation and scheduling is computed by ILP equation solving, and is obviously unknown at the time of setting up the ILP equations. The proposed solution is to introduce a 0/1 variable $w_{i,j}$ for each pair i and j of fixed targets to describe the UAV that successively handles the two targets (one immediately after the other). If the variable is one then target i is handled before j. Otherwise, the variable is zero. In addition, the same UAV must handle both tasks.

The flight time T_{fly} to a fixed target j is defined as follows:

$$T_{j,fly} \propto \sum_{\forall target_i} w_{i,j} Dist(target_i, target_j) \times (\sum_{k \in UAVs} x_{i,k} x_{j,k}) \tag{4.6}$$

The next constraint expresses the fact that each task is handled by one UAV after the UAV handles exactly one task (with the exception of the "dummy" start node), so for each task i:

$$\sum_{\forall target_j} w_{i,j} = 1 \tag{4.7}$$

4.4.6 UAV collaboration

In collaboration, the identity of the collaborating UAVs and the nature of the activities involved in collaboration is not known for centralized decision making, but instead is decided during UAV operation. The centralized decision making assigns and schedules tasks so that the flexibility of collaboration (if needed) is maximized.

As shown in Figure 4.4, a UAV might decide to collaborate after each of the activities related to a task, such as the fly, detect, and handle activities. The flexibility for collaboration depends on the slack time between the end of the current activity and the beginning of the next activity, and the deadline of the target handling for

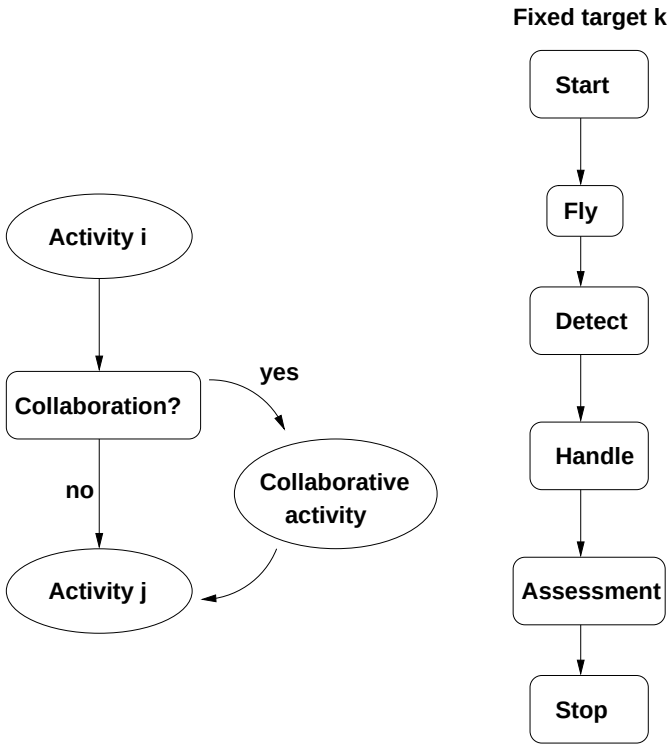

Figure 4.6: Modeling for dynamic collaboration

which the collaborative action is requested. The more overlapping exists between the slack time and the deadline the more flexibility exists in collaborating to meet the deadline (e.g., the deadline is before the starting of the slack time, or after the end of the slack time) then there is no possibility of the UAV to participate in handling the target.

Figure 4.6 is used to explain the ILP equations for collaboration. For each target k, we define $SetC_k$ as the set of targets for which the assigned UAVs are candidates for collaboration. $SetC_k$ can be set statically based on the geographical proximity of the targets (this information is known), or can be decided dynamically at run time depending on the current slack time of a UAV, and hence its flexibility to fly to more distant targets without violating the deadlines of its assigned targets. In this

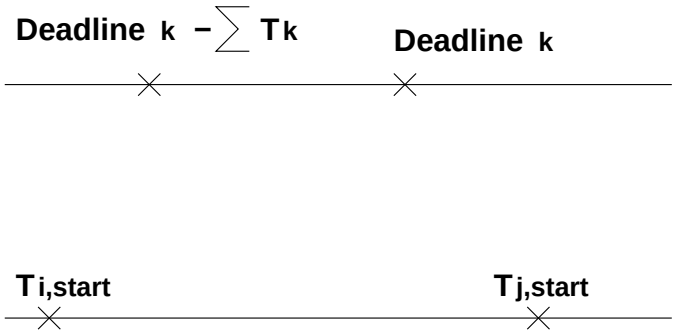

Figure 4.7: Flexibility in collaboration

discussion, $SetC_k$ is assumed to be static.

The flexibility in participating to a collaborative handling of target k between activities i and j (scheduled in this order) is proportional to the following value:

$$Fl_{i,j,k} \propto [Activity_{i,end}, Activity_{j,start}] \cap [(Deadline_k - \sum T_k), deadline_k] \quad (4.8)$$

Variables $Activity_{i,end}$ and $Activity_{i,start}$ are the end time of Activity i and the start time of Activity j. $[Activity_{i,end}, Activity_{j,start}]$ represents the time interval defined by the two moments. $Deadline_k$ is the deadline set for target k, and $\sum T_k$ is the time required to perform all activities related to target k, e.g., detect, handle, and assess.

Figure 4.7 illustrates the definition of the flexibility constraint for UAV collaborating on the handling of target k. Targets i and j are allocated to the UAV. The condition for collaboration is defined by the following equations:

$$T_{i,end} \leq Deadline_k - \sum T_k \quad (4.9)$$

$$Deadline_k \leq T_{j,start} \quad (4.10)$$

The equation for the flexibility for UAV m is shown in equation 4.11. Where $x_{m,i}$ and $x_{m,j}$ are both equal to one, it means the same UAV m are handling $Activity_i$ and $Activity_j$. $x_{m,k}$ is required to equal to zero, which means UAV m is not initially assigned to $Activity_k$, but UAV m plans to handle $Activity_k$ on account of flexibility Fli, j, k's calculation.

$$Fl_{i,j,k} \propto x_{m,i}x_{m,j}(1 - x_{m,k})(Deadline_k - T_{i,end})(T_{j,start} - Deadline_k) \tag{4.11}$$

The total flexibility of an UAV to participate to collaborative activities is:

$$Fl_{i,j} = \sum_{k \in SetC_k} Fl_{i,j,k} \tag{4.12}$$

The overall cost function includes a term to maximize the flexibility of UAVs participating to collaborative activities.

4.4.7 Pop-up targets

Each pop-up target has a specific probability of appearing in a given 3D area, and remains visible for a time which is also defined as a probability over a certain interval. The proposed modeling assumes that the target has equal probability of appearing in a known volume, and similarly, the time of being visible is a uniform probability over a given interval.

A main goal of centralized decision making is to maximize the chances of handling as many pop-up targets as possible. The chances of a pop-up target becoming visible depends on the constant probability as well as the time associated with the preemptive flight mode, during which the UAV can handle a pop-up target (see Figure 4.5). Assuming that time T_{X_i} is the time when point X_i (the beginning of the preemptive flight segment) is reached, and time T_{X_f} is the time when point X_f (the end of the

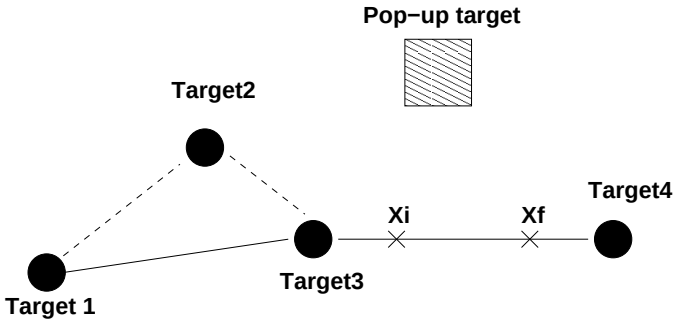

Figure 4.8: 0/1 variables for UAV collaboration modeling

preemptive flight segment) is traversed by the UAV. The chances of the pop-up target PP_i appearing during that time interval is as follows:

$$p_{PP_i} = (T_{X_f} - T_{X_i}) prob_{PP_{i,appearance}} \tag{4.13}$$

Thus, increasing the chances of handling a pop-up target involves increasing the time interval defined by the time moments T_{X_i} and T_{X_f}. This is obviously correlated to the decisions of assigning fixed targets to UAVs, and then scheduling the targets. For example, in Figure 4.8, increasing the chances of handling the pop-up target might require deciding that $Target_2$ is not handled by the UAV in order to increase the length of the time interval $T_{X_f} - T_{X_i}$, while still meeting the deadline constraints associated with $Target_4$.

Using Figure 4.8 as an illustrative example, the expressions for the time moments T_{X_i} and T_{X_f} are defined as follows:

$$T_{X_i} = t_{Target_{3,end}} + \frac{|X_i - X_{Target_3}|}{speed_{UAV}} \tag{4.14}$$

$$t_{Target_{4,start}} = T_{x_f} + \frac{|X_{Target_4 - X_f}|}{speed_{UAV}} \tag{4.15}$$

and

$$T_{X_f} \leq T_{X_i} + \frac{|X_f - X_i|}{speed_{UAV}} \tag{4.16}$$

$speed_{UAV}$ is a known constant in the equations. Also, the relationship includes an inequality (equation 4.16) as the UAV might travel at a lesser speed than the value $speed_{UAV}$ in order to increase the chances of the pop-up target being observed by the UAV.

The time required for the UAV to cover the segment between targets $Target_i$ and $Target_j$, including the handling of a pop-up target is as follows:

$$T_{flight} = T_{non\_pr,1} + T_{preempt} + T_{non\_pr,2} \tag{4.17}$$

where the variables $T_{non\_pr,1}$ and $T_{non\_pr,2}$ are the times required for the non-preemptive zone at the beginning and at the end of the segment (see Figure 4.5), and variable $T_{preempt}$ represents the time of preemptive flight.

$$T_{non\_pr,1} = \frac{|X_i - X_{Target_i}|}{speed_{UAV}} \tag{4.18}$$

$$T_{non\_pr,2} = \frac{|X_{Target_j} - X_f|}{speed_{UAV}} \tag{4.19}$$

The following equation is valid, where values T_{Detect}, T_{Handle} and $T_{Assessment}$ are constant:

$$T_{preempt} = T_{fly} + T_{Detect} + T_{Handle} + T_{Assessment} \tag{4.20}$$

The value of variable T_{fly} is the time it takes to the UAV to fly to a visible pop-up target. This time obviously depends on the position of the pop-up target, which is random inside a given 3D region. For the centralized decision making step, T_{fly} is estimated to be the average of the flight time assuming that the pop-up target has equal probability of appearing inside the 3D area of volume V.

$$T_{fly} = \frac{1}{speed_{UAV}} \frac{\int_V distance(X)dX}{V} \tag{4.21}$$

where $distance(X)$ represents the distance from an arbitrary point X inside the volume V to the line defined by $Target_i$ and $Target_j$.

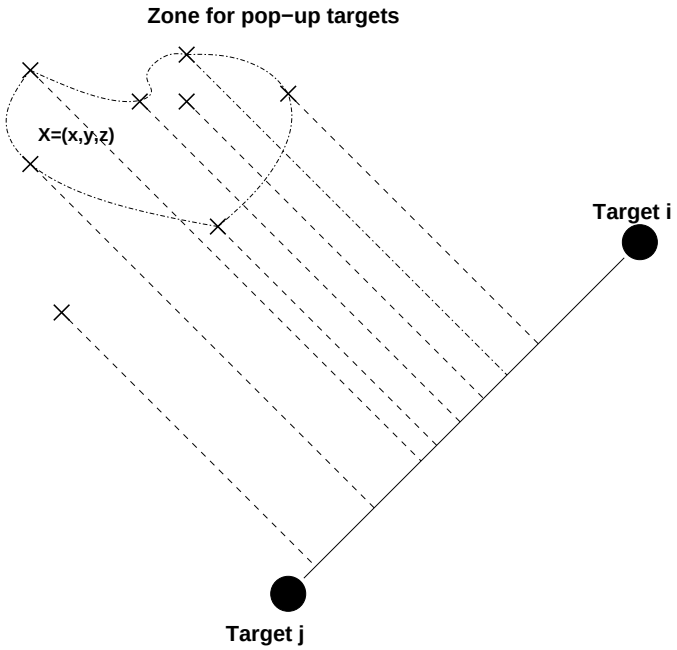

Figure 4.9: Average distance to pop-up target

Figure 4.9 presents the case in which the volume V of a pop-up target's zone cannot be predicted. Instead, the pop-up target can emerge as a set of points, which are labelled with 'x' in the figure. In this case, the flight time to a pop-up target is estimated as follows:

$$T_{fly} = \frac{1}{speed_{UAV}} \frac{\sum_{i \in X} distance(X_i)}{Card(X)} \tag{4.22}$$

X is the set of all points in which the pop-up target can be placed, and X_i is one of the points. $Card(X)$ is the cardinality of the set X.

4.4.8 Cost function

The cost function is a weighted sum that expresses the goals of (i) minimizing the cumulated penalties for exceeding the predefined deadlines for the static targets, and (ii) maximizing the probability for handling pop-up targets:

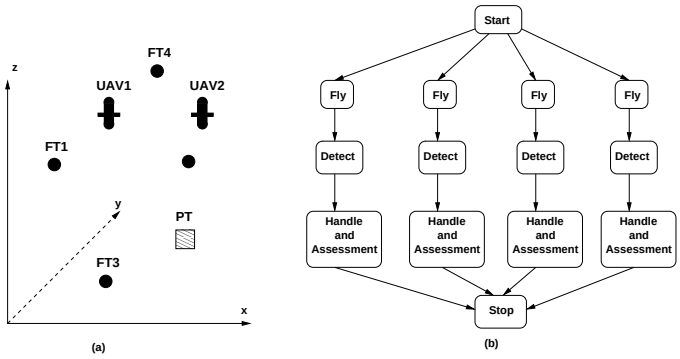

Figure 4.10: Case Study

$$Cost = \alpha \times \sum_{\forall targets_i} (T_{i,end} - T_{i,deadline})^2 + \beta \times \sum pPP_i + \gamma \times \sum_{\forall targets_{i,j}} Fl_{i,j} \quad (4.23)$$

4.5 Case Study

This section presents the ILP equations for the case study in Figure 4.10. The case study includes two UAVs, four targets (FT) and one pop-up target (PT) positioned between targets 2 and 3. Figure 4.10(b) shows the task graph for this application. Each of the four threads corresponds to one of the fixed targets. The model was developed for *lp_solve* ILP solver or SNOPT solver [52].

4.5.1 Constraint 1, individual task start and end time

The following equations define the task start and end times for tasks Fly (index 2), Detect (index3), Handle and Assessment (index 4) of the first thread:

$$T1 \leq T2\_start \quad (4.24)$$

$$T2\_end = T2\_start + T2_{ex,1}x2\_1 + T2_{ex,2}x2\_2 \quad (4.25)$$

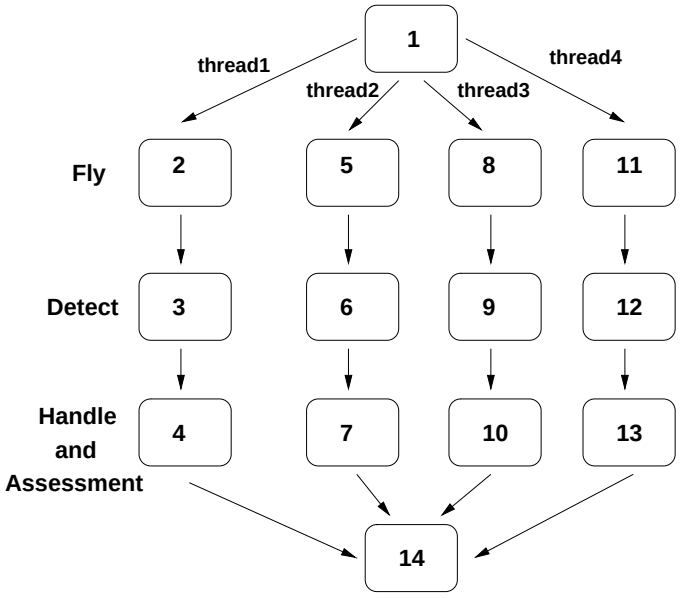

Figure 4.11: Case Study, Index Representation

$$x2\_1 + x2\_2 = 1 \tag{4.26}$$

$$T2\_end \leq T3\_start \tag{4.27}$$

$$T3\_end = T3\_start + T3_{ex,1}x3\_1 + T3_{ex,2}x3\_2 \tag{4.28}$$

$$x3\_1 + x3\_2 = 1 \tag{4.29}$$

$$T3\_end \leq T4\_start \tag{4.30}$$

$$T4\_end = T4\_start + T4_{ex,1}x4\_1 + T4_{ex,2}x4\_2 \tag{4.31}$$

$$x4\_1 + x4\_2 = 1 \tag{4.32}$$

$T_{i_{ex,j}}$ is the time it takes UAV j to perform task i. Variables xi_k are 0/1 variables with value 1 if task i is allocated to UAV k, and value 0 otherwise. Similar equations were introduced for all the activities or tasks in the graph 4.10 and 4.11.

4.5.2 Constraint 2, define the flight time between successive targets

The following equations define the flight time between successive targets depending on the allocation of targets to UAVs and the order in which threads are visited. Such equations are introduced between each possible sequencing of the two targets i and j. The 0/1 variables zi_j are 1, if thread i is executed before thread j, thus target i is before target j.

$$TF1 \geq d21 + z2\_1T_\infty + x2\_1T_\infty + x5\_1T_\infty - 3T_\infty \tag{4.33}$$

$$TF1 \geq d31 + z3\_1T_\infty + x2\_1T_\infty + x8\_1T_\infty - 3T_\infty \tag{4.34}$$

$$TF1 \geq d41 + z4\_1T_\infty + x2\_1T_\infty + x11\_1T_\infty - 3T_\infty \tag{4.35}$$

dij is the time it takes a UAV to fly the distance between target i and j at a constant average speed. T_∞ is a very large value. The reason for giving both $x5\_1$ and $x2\_1$ in the same equation 4.33 is to guarantee that target 2 and 1 are both handled by the same UAV 1. The same requirements apply to $x8\_1$ and $x11\_1$ for thread 3 and 4 separately. If xi_1 is changed to xi_2, or the second index to 2, the other three corresponding constraints are obtained for targets handled by UAV 2. (Equation 4.36 – 4.38)

$$TF1 \geq d21 + z2\_1T_\infty + x2\_2T_\infty + x5\_2T_\infty - 3T_\infty \tag{4.36}$$

$$TF1 \geq d31 + z3\_1T_\infty + x2\_2T_\infty + x8\_2T_\infty - 3T_\infty \tag{4.37}$$

$$TF1 \geq d41 + z4\_1T_\infty + x2\_2T_\infty + x11\_2T_\infty - 3T_\infty \tag{4.38}$$

4.5.3 Constraint 3, define tasks' scheduling between successive threads

The next set of equations is defined for each pair of tasks of the four parallel threads. They indicate that a UAV can execute a single task for each of the two threads at consecutive times, such as task assessment (index 4) of thread 1 and fly (index 5) of thread 2 (Fig. 4.10), and task assessment (index 7) of thread 2 and fly (index 2) of thread 1 in this case as shown next. The 0/1 variables zi_j are 1 if thread i is performed before thread j. Otherwise, if zi_j are 0 when thread i is performed after thread j. T_∞ is a very large constant. Equation 4.39 to 4.43 are tasks' scheduling between thread 1 and 2. In this example, for the 4 threads case, it contains $C_4^2 = 6$ different pairwise zi_j. Therefore there are a total 24 tasks' scheduling constraints.

$$T4\_end \leq T5\_start + T_\infty z2\_1 + T_\infty - \frac{T_\infty}{2}x4\_1 - \frac{T_\infty}{2}x5\_1 \tag{4.39}$$

$$T4\_end \leq T5\_start + T_\infty z2\_1 + T_\infty - \frac{T_\infty}{2}x4\_2 - \frac{T_\infty}{2}x5\_2 \tag{4.40}$$

$$T7\_end \leq T2\_start + T_\infty z1\_2 + T_\infty - \frac{T_\infty}{2}x7\_1 - \frac{T_\infty}{2}x2\_1 \tag{4.41}$$

$$T7\_end \leq T2\_start + T_\infty z1\_2 + T_\infty - \frac{T_\infty}{2}x7\_2 - \frac{T_\infty}{2}x2\_2 \tag{4.42}$$

$$z1\_2 + z2\_1 = 1 \tag{4.43}$$

4.5.4 Constraint 4, the flexibility of UAVs for collaboration

The following equations define the flexibility of UAVs to collaborate. Variable $Fl_{k\_i\_j}$ is the flexibility of UAV collaboration in handling thread k (target k) while performing thread i (target i) and j (target j). And $Fl_{k\_i\_j}$ also means thread i is handled before j. For instance, $Fl_{3\_1\_2}$ is the flexibility of UAV collaboration in handling thread 3 while performing thread 1 and 2 originally.

$$Fl_{3\_1\_2} \leq -(Deadline3 - T4\_end)(T5\_start - Deadline3) + T_\infty - \frac{T_\infty}{4} z1\_2 - \\ - \frac{T_\infty}{4} x4\_1 - \frac{T_\infty}{4} x5\_1 - \frac{T_\infty}{4}(1 - x8\_1) \tag{4.44}$$

$$Fl_{3\_1\_2} \leq -(Deadline3 - T4\_end)(T5\_start - Deadline3) + T_\infty - \frac{T_\infty}{4} z1\_2 - \\ - \frac{T_\infty}{4} x4\_2 - \frac{T_\infty}{4} x5\_2 - \frac{T_\infty}{4}(1 - x8\_2) \tag{4.45}$$

Deadlinek is the deadline for target k. Equation 4.44 and 4.45 are both for flexibility $Fl_{3\_1\_2}$, the only difference is equation 4.44 expresses that both target 1 and 2 are preassigned to UAV1, target 3 is originally not assigned to UAV1. Only in this way does target 3 have the flexibility to reassigned to UAV1 between target 1 and 2. For equation 4.45, all the conditions are the same except target 1 and 2 are now preassigned to UAV2, and target 3 is originally not assigned to UAV2. So target 3 has the flexibility to reassigned to UAV2 between target 1 and 2. Similar relationships exist for all the triplets of targets.

Knowing how to calculate triplets targets allows the addition of this new group of constraints into the objective function. Thus, the fixed target scheduling with optimized flexibility can be selected. It is one way for collaborating UAVs targets assignment and scheduling.

4.5.5 Constraint 5, the flexibility of UAVs for handling a pop-up target

The flexibility of a UAV in handling the pop-up target positioned between the fixed target 2 and 3 is given as follows.

$$popup \leq -(PR)(T8\_start - T7\_end) + T_\infty - \frac{T_\infty}{3} z2\_3 - \\ - \frac{T_\infty}{3} x8\_1 - \frac{T_\infty}{3} x7\_1 \tag{4.46}$$

$$popup \leq -(PR)(T5\_start - T10\_end) + T_\infty - \frac{T_\infty}{3} z3\_2 - \\ - \frac{T_\infty}{3} x5\_1 - \frac{T_\infty}{3} x10\_1 \tag{4.47}$$

PR in the above equations 4.46 and 4.47 is the uniform probability of the pop-up target to appear.

1. **Explanation for handling pop-up targets**

   From equation, it is seen that the larger the time interval between target 2 and 3, the higher the probability for a UAV to handling the pop-up target. Equation 4.46 is for $z2\_3$ equals to 1, $x8\_1$ and $x7\_1$ are equal to each other. Or in the other words, target 2 is handled before 3, and target 2 and 3 are handled by the same UAV. For equation 4.47, $z3\_2$ equals to 1, $x5\_1$ and $x10\_1$ are asked to be equal. Thus target 2 is handled after 3. Still both target 2 and 3 are handled by the same UAV. If any one of above requirements fails, $|popup|$ is not maximized.

2. **Limitation for handling pop-up targets**

   For some unknown reason, the non-linear solver (SNOPT) could not solve situations that involve handling pop-up targets properly. One possible explanation is the occurrence of a negative value in the objective function. An additional problem occurs when there is more than one pop-up target, the number of variables increase dramatically. The number of constraints also increases. These non-linear solver's limitations are discussed briefly in Section 5.2.

4.5.6 Final cost function to minimize

The cost function to be minimized is as follows:

$$min : T14 + \sum_{i,j,k=1...4} Fl_{i\_j\_k} + POPUP \tag{4.48}$$

The first term $T14$ is the T_end time for all 4 targets.(Fig. 4.11) $T14$ should satisfy the following 4 inequity equations.

$$T4\_end \leq T14 \tag{4.49}$$

$$T7\_end \leq T14 \tag{4.50}$$

$$T10\_end \leq T14 \tag{4.51}$$

$$T13\_end \leq T14 \tag{4.52}$$

$T4$, $T7$, $T10$ and $T13$ are the end time of thread 1, 2, 3 and 4 separately in Figure 4.11. Therefore, the first term $T14$ is the completion time of the four threads to be close to the fixed deadline. This captures the timing constraints. The second term minimizes variables $Fl_{i\_j\_k}$. Since their values are negative, it actually maximizes the flexibility of UAV collaboration. The last term $POPUP$ maximizes the probability of handling pop-up targets as the value of $POPUP$ is always negative.

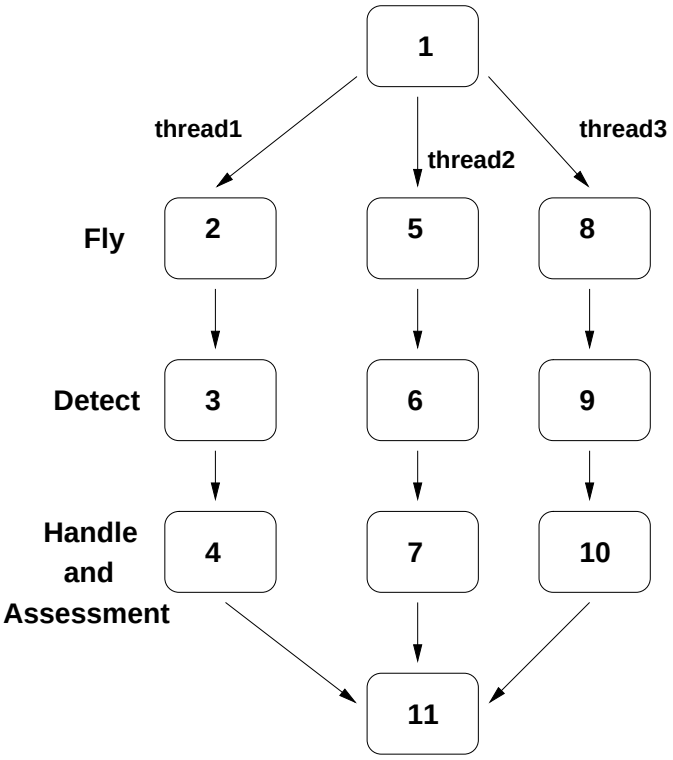

Figure 4.12: Index Representation for Thread-3 Example

4.5.7 A simple example to calculate flexibility

As illustrated in Figure 4.12, suppose there are three fixed targets (FT) and two UAVs. The figure shows the task graph for this specific case, each of the three threads corresponds to one of the fixed targets.

The constraints group can be easily structured from 4.5.1 to 4.5.4. All these constraints are expressed in mixed-integer linear programming format, with 0/1 integer variables $x_{i,j}$ and $z_{i,j}$. For simplicity reasons, there are some constants inside the constraint group, such as the fly time between differen threads or cities. The model thus can be solved by the non-linear solver $SNOPT$. One of the possible time assignment results is given in table 4.1.

From table 4.1, we can see that both of the fixed threads 1 and 2 are handled by the

Table 4.1: Time assignment for different thread/cities

Thread Index	1	2	3
Start	0	100	0
End	30	140	30
Deadline	50	200	50
UAV_i	1	1	2

same UAV 1, and fixed thread 3 is handled by the other UAV 2. To calculate UAVs' flexibility for collaboration as explained in 4.5.4, flexibility table 4.2 is obtained.

The reason some flexibility values are not applicable is because the two original targets are handled by different UAVs. Fox example, $Fl1\_2\_3$. $Fl1\_2\_3$ means putting fixed target 1 between target 2 and 3. From table 4.1, $x5\_1 = 1$, and $x8\_2 = 1$, which means target 2 is handled by UAV 1, while target 3 is assigned to UAV 2. Thus, $Fl1\_2\_3$ is not applicable (NA).

From equation 4.53 and Figure 4.13, it is clear that $Fl_{3\_1\_2} = 1000$. Thus, the fixed target 3 has the flexibility of 1000 to be handled by UAV 1. Originally target 3 was assigned to UAV 2. By doing this rearrangement and cooperation, UAV 2 can be saved for other uses. Therefore, this thread-3 task assignment and scheduling for collaborating UAVs is successful. For the complete constraints group, please see Appendix B.

$$Fl_{3\_1\_2} = (Deadline3 - T4\_end)(T5\_start - Deadline3) = (50-30)(100-50) \quad (4.53)$$

Table 4.2: Flexibility for thread-3 example

Thread Index	Thread-3
T11	140
$Fl1\_2\_3$ $Fl1\_3\_2$	NA
$Fl2\_1\_3$ $Fl2\_3\_1$	NA
$Fl3\_1\_2$ $Fl3\_2\_1$	$Fl3\_1\_2$ $= 1000$

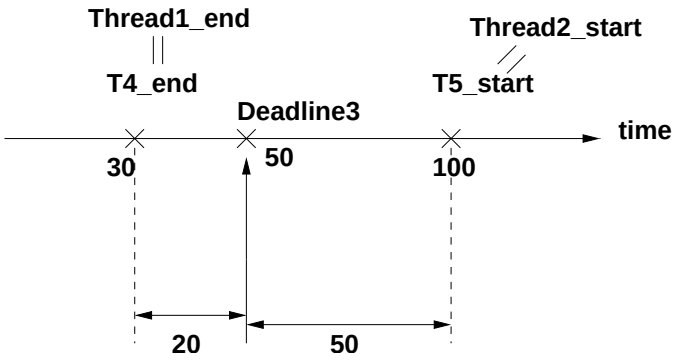

Figure 4.13: Time axis for $Fl_{3\_1\_2}$

4.6 Conclusion

This chapter describes the problem of Unmanned Autonomous Vehicles (UAVs) collaboration for target handling as a task graph scheduling and assignment problem. The chapters offers a mathematical modeling of the problem based on Mixed Integer Nonlinear Programming. The formulation can be tackled using state-of-the-art solvers, or can be used as a starting point for developing optimization heuristics. Each target handling activity is a parallel thread consisting of the four consecutive processes: UAV flying activity, target detection process, handle, and assessment process. The novelty is in developing a formal description of multi-UAV collaboration as compared to similar work which mostly focuses on single UAV control.

For task graph scheduling and assignment, the proposed methodology is based on the same solving technique (Mixed Integer Nonlinear Programming) as the algorithms in Chapter 3 (hierarchical mixed integer programming based algorithm for floorplanning in VLSI design). The main step of the methodology includes allocation and scheduling of fixed-position targets to individual UAVs, computing the time intervals for collaborative behavior, and finding the regions within which pop-up target can be safely handled by an UAV without violating timing constraints. We also put forth the idea of decentralized UAV control, which uses the allocation, scheduling, and constraints information for controlling the reactive behavior of each UAV.

For a case study with two UAVs, four targets and one pop-up target, the chapter discusses the constraints and the cost function for the Mixed Integer Nonlinear Programming description. After we input the constraints and the cost function into the integer linear/nonlinear program solver SNOPT, we compute the optimum task scheduling and assignment result for the two UAVs. We also calculate the UAV flexibility to handle new pop-up targets. Results of this case study displayed that integer linear programming is a good optimization framework for extending continuous linear

programming problem to include binary or integer decision variables for solving the UAV collaboration problem. These variables can be used to model logical constraints such as obstacle and targets' deadline avoidance rules.

MINLP modeling for collaborating UAVs is a very recent research topic. There are many potential research directions for this work. For example, SNOPT cannot handle large number of variables and constraints. Therefore, if we want to tackle a large application with many threads and UAVs, or if we want to obtain a real-time trajectory optimizer, which includes more environment-related unknowns, we must build a new solver. It needs to combine heuristic optimization methods, such as Genetic Algorithm(GA), Simulated Annealing(SA) algorithm etc, with MINLP method for offering an optimal yet fast solution.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This dissertation work focuses on three main topics. First, it analyzes the topological structure of large size electronic circuits. Secondly, it proposes a hierarchical mixed integer programming based floorplanning algorithm that attempts to exploit the fact that certain VLSI networks may have structural patterns. Finally, to further extend integer linear programming(ILP) method for optimization, we propose a Mixed Integer Nonlinear Programming (MINLP) based algorithm for task assignment and scheduling for collaborating Unmanned Autonomous Vehicles (UAVs). Hence, the common challenge behind the three main topics is the problem of using MINLP based algorithms for solving various optimization problems. Specifically, the algorithm for electronic VLSI circuit floorplanning is realized by integer non-linear programming, and the problem of collaborating UAVs is handled by integer linear programming.

The dissertation's first part presents the new idea of studying VLSI electronic circuits using approaches originated from the area of complex systems. Through experiments, we showed that large size VLSI electronic circuits have broad scale-free patterns. We computed their power-law degree distribution in a range, and we further

compared it with existing scale-free network models, in particular BA model. The proposed triad formation (TF) model explains how to generate a large clustering coefficient inside large-size electronic circuits. At the same time, the coexisting of broad scale-free patterns for degree distribution and large clustering coefficients suggests that there may exist some clusters or motifs inside the circuits' network. Understanding these particular electronic circuit topologies structure can help the tedious optimization work during floorplanning and placement in VLSI physical design. For instance, if we identify that there are many cliques (fully connected clustered graph) inside one block, we can exploit this knowledge by finding and using the cliques as basic cells in floorplanning and placement without handling the nodes inside a cell (as in traditional methods), because all the wire length inside a clique will remain the same for different cell placements. Also, if we perform floorplanning using a partition-based hierarchical algorithm, like our new floorplanning algorithm Hercules, the partition size K could be set-up a-priori, so that K is equal with the number of clusters at each level. By doing this, the minimal cut size between different partitions is reduced. Therefore, the final performance (e.g., total wire length) is also improved.

As the second contribution of the thesis, we developed a hierarchical Mixed Integer Nonlinear Programming based algorithm for floorplanning in VLSI design. The algorithm is called Hercules. The method is based on several aspects. Inside the top-down structure, we first use hMETIS partitioning method to tackle the large size of a circuit by building a hierarchical partitioning structure at different levels. Within each partitioning level, the finer floorplanning problem is solved by SNOPT guided by simulated annealing (SA) algorithm which sets the relative positioning constraints for the floorplanner. After taking the external connections into account, terminal propagation (TP) method is performed. Experimental results on standard benchmark examples (MCNC benchmark) show that the proposed method offers results of similar quality in a much shorter time than modern floorplanning algorithms.

The third topic is an extended application of Mixed Integer Nonlinear Programming method to predictive collaborative control of UAVs (Unmanned Autonomous Vehicles) in partially known environments. The MINLP model for fixed targets assigning and scheduling was proposed. The work also studied UAVs flexibility in handling pop-up targets. MINLP models are also solved using SNOPT, the same solver package we applied in the previous floorplanning algorithm. UAVs collaborative performance for a higher number of pop-up targets, or with more unknown environment parameters, and real-time trajectory optimizer are still open questions.

5.2 Future Work

So far my research has focused on complex network structure, its application to large size electronic systems, such as floorplanning algorithms in SoC (System on Chip) physical design procedure, and target/task assignment and scheduling for collaborating UAVs. In order to achieve the broad goals enumerated in Chapter 1, following topics need to be further studied. I currently see these directions to improve the work presented in this dissertation.

1. **Deeper Relationship Between Circuits and Complex Networks**

   The discovery of broad-scale patterns and large clustering coefficient is the first characteristic we find in the electronic circuit structure analysis. In complex network context, there are more intensive relationships between the topology and the dynamics of complex networks. Is there any similar relationship between circuit building blocks and circuit evolving networks? Does the top-down hierarchical partitioning process still satisfy complex networks' properties? How to differentiate network structures with the same statistical information, such as degree distribution, clustering coefficient and minimal distance? All these

questions are intriguing.

2. **Hierarchical Partitioning and Clustering Identification**

   Hierarchical partitioning is conceptually a top-down procedure because it looks at the entire circuit as a basic entity. In contrast, clustering identification is a bottom-up procedure since it brings highly connected nodes together. Thus, clustering looks at the individual nodes and their connections as basic entities. The previous chapters showed that we probably could reduce the wire length by considering a combination of partitioning and clustering in a same algorithm. To find the best approach one has to check the inherent nature of the target circuits at a certain point during the top-down process. If the graph begins to show signs of isolated clusters, then partitioning may disturb this structure if the chosen partitioning size is not reasonable. For instance, if there are four individual clusters and the chosen partitioning size is larger than four, the inherent nature of the graph is broken and the cut sets will be increased. Therefore, clustering may not always be the right choice if all nodes are highly connected to each other. Highly connected nodes lead to the formation of one single cluster, which includes all the nodes. This formation defeats the purpose of breaking up the circuit. There are some work incorporated hierarchical clustering in the algorithm [42] [43], but the complete solution for any clustering problem is still very difficult. It is also a challenge research topic in graph theory in mathematics.

3. **Solver Feasibility Study**

   In floorplanning algorithm in Chapter 3, and tasks' assignment and scheduling for collaborating UAVs in Chapter 4, the non-linear solver SNOPT showed some limitations, especially when we handled a large number of variables. Therefore,

the topic about how to merge and reduce different variables and constraints in MILP/MINLP is worth to study more. As we mentioned in Chapter 4's conclusion, the best way to solve UAVs' collaboration with more number of unknown environment parameters is to create our own integer linear program solver combined with new optimization methods.

4. **UAVs Real-Time Trajectory Optimizer**

   Many of UAVs' tasks are preplanned using reconnaissance or environment information. Because of the solver's (SNOPT) limitation, it is very difficult to include all the unknown environment parameters. For example, pop-up targets and a real-time trajectory simulation are still missing in the experiment. If we could combine some new optimization methods, such as Genetic Algorithm (GA), Simulated Annealing (SA) or other heuristic models with the non-linear solver (SNOPT), the UAVs' task planning flexibility and optimized trajectory solutions could be obtained more thoroughly.

Bibliography

[1] S. H. Strogatz, "Exploring complex networks", *Nature*, vol. 410, pp. 268-276, March 2001.

[2] P. Erdös and A. Rényi, "On the evolution of random graphs", *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17-60, 1959.

[3] S. Milgram, "The small-world problem", *Psychology Today,* vol. 2, pp. 60-67, 1967.

[4] Albert-László Barabási, Réka Albert, "Emergence of Scaling in Random Networks", *Science*, vol. 286, pp. 509-512, Oct 1999.

[5] Albert-László Barabási, Réka Albert, Hawoong Jeong, "Mean-field theory for scale-free random networks", *Physica A*, vol. 272, pp. 173-187, July 1999.

[6] Réka Albert, Albert-László Barabási, "Statistical mechanics of complex networks", *Reviews of Modern Physics*, vol. 74, Jan 2002.

[7] Cancho *et al*, "The Topology of Technology Graphs: Small World Patterns in Electronic Circuits", *Phys. Rev. E*, vol.64, 046119, Sept 2001.

[8] Kott, A., *Advanced Technology Concepts for Command and Control*, Xlibris Corp., Philadelphia, PA, 2004.

[9] Samad, T., and Balas, G., *Software-Enabled Control: Information Technology for Dynamical Systems*, Wiley/IEEE Press, Hoboken, NJ, 2003.

[10] Butenko, S., Murphey, R., and Pardalos, P., *Recent Developments in Cooperative Control and Optimization*, Kluwer Academic, Norwell, MA, 2003.

[11] Grundel, D., Murphey, R., and Pardalos, P., *Theory and Algorithms for Cooperative Systems*, vol. 4, Series on Computers and Operations Research, World Scientific, Hackensack, NJ, 2004.

[12] Floudas, C. A., *Nonlinear and Mixed-Integer Programming Fundamentals and Applications*, Oxford Univ. Press, Oxford, England, UK, 1995.

[13] D.J. Watts, S.H. Strogatz, "Collective dynamics of 'small-world' networks", *Nature*, vol. 393, pp. 440, 1998.

[14] A. Barrat, M. Weigt, "On the properties of small-world network models", *Eur. Phys. J. B*, vol. 13, pp. 547-560, 2000.

[15] Kwang *et al*, "Classification of scale-free networks", *PNAS*, vol. 99, no. 20, 12583-12588, 2002.

[16] Douglas B. West, "Introduction to graph theory", *Prentice Hall*, 2001.

[17] Petter Holme, Beom Jun Kim, "Growing Scale-Free Networks with Tunable Clustering", *Phys. Rev. E*, 2002.

[18] R. Albert, H.Jeong, & A.-L. Barabási, "Diameter of the World-Wide Web", *Nature*, 400, pp. 130-131, 1999.

[19] M. Faloutsos, P. Faloutsos, & C. Faloutsos, "On Power-Law Relationships of the Internet Topology", *ACM SIGCOMM99, Rev.29*, pp. 251-260, 1999.

[20] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, & A.L.Barabási, "The large-scale organization of metabolic networks", *Nature* , 607, 651, 2000.

[21] H. Jeong, S.P. Mason, S.N. Oltvai, & A.-L.Barabási, *Nature*, London.411, 41, 2001.

[22] R. Milo, S. Shen-Orr *et al*, *Science*, vol. 298, pp. 824-827, 2002.

[23] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning", *VLSI Design*, Vol.11, No.3, pp. 285-300, 2000.

[24] S. Sutanthavibul, E. Shragowitz and J. Ben Rosen, "An analytical approach to floorplan design and optimization", *Proc. 27th ACM/IEEE Design Automation Conference*, 1990.

[25] Stanford Business Software, *http://sbsi-sol-optimize.com*, 2006.

[26] N. Sherwani, "Algorithms for VLSI physical design automation", *Kluwer Academic Publishers*, 1995.

[27] Gigascale Systems Research Center, *http://www.gigascale.org/bookshelf/*, 2006.

[28] R. Otten, "Efficient floorplan optimization", in*Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 499-502, 1983.

[29] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design", in *Proc. 23rd ACM/IEEE Design Automation Conf.*, pp. 101-107, 1986.

[30] Temo Chen and Micheal K. H. Fan, "On convex formulation of the floorplan area minimization problem", in*Proc. Int. Symp. Physical Design*, pp. 124-128, 1998.

[31] P. Chen and E.S.Kuh, "Floorplan sizing by linear programming approximation", in*Proc. 37th Design Automation Conf.*, pp. 468-471, 2000.

[32] H. Murata, K. Fujiyoshi and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair",*IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1518-1524, Dec. 1996.

[33] H. Murata and E.S.Kuh, "Sequence-pair based placement method for hard/soft/preplaced modules", in*Proc. Int. Symp. Physical Design*, pp. 167-172, 1998.

[34] J. Kim and Y. Kim, "A linear programming-based algorithm for floorplanning in VLSI design",*IEEE Trans. Computer-Aided Design*, vol. 22, pp. 584-592, May. 2003.

[35] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, "Module placement on BSG-structure and IC layout applications", in*Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 484-491, 1996.

[36] M. Kang and W. Dai, "General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure", in*Proc. Asia South Pacific Design Automation Conf.*, pp. 265-270, 1997.

[37] S. Sutanthavibul, E. Shragowits and J.Ben Rosen, "An analytical approach to floorplan design and optimization", in*Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 187-192, 1990.

[38] A. Dunlop and B. Kernighan, "A procedure for placement of standard-cell VLSI circuits", *IEEE Trans. Computer-Aided Design*, vol. 4, pp. 92-98, Jan. 1985.

[39] F. Y. Young, D. F. Wong and H. H. Yang, "Slicing floorplans with boundary constraints", *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 1385-1389, Sept. 1999.

[40] F. Y. Young and D. F. Wong, "Slicing floorplans with range constraint", *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 272-278, Feb. 2000.

[41] F. Y. Young, C. C. N. Chu, W. S. Luk, and Y. C. Wong, "Handling soft modules in general nonslizing floorplan using lagrangian relaxation", *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 687-692, May 2001.

[42] T. Yamanouchi, K. Tamakashi, and T. Kambe, "Hybrid floorplanning based on partial clustering and module restructuring", in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 478-483

[43] S. Dongen, "A clustering algorithm for graphs", *Technical Report, Stichting Mathematich Centrum*, 2000.

[44] M. Sarrafzadeh and C. K. Wong, "An introduction to VLSI physical design", *McGraw-Hill companies, Inc.*, 1996.

[45] N. Sherwani, "Algorithms for VLSI physical design automation", *3rd Edition Kluwer Academic Publishers*, 1999.

[46] R. Mayo, M. Arnold, W. Scott, D. Stark and G. Hamachi, "1990 DECWRL/Livermore Magic Release", *Technical Report, Western Research Laboratory*, 1990.

[47] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions", *IEEE/ACM, Design Automation Conference*, 1982.

[48] A. Ranjan, K. Bhazargan, M. Sarrafzadeh, "Fast hierarchical floorplanning with congestion and timing control", *Proc. ICCAD00*, 2000.

[49] A. Ranjan, K. Bhazargan, S. Ogrenci, M. Sarrafzadeh, "Fast floorplanning for effective prediction and construction", *IEEE Transaction on VLSI Systems*, vol. 9, No. 2, April, 2001.

[50] H. Yamazaki, K. Sakanushi, and Y. Kajitani, "Optimum packing of convex-polygons by a new data strcture sequence-table", in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, 2000, pp. 821-824

[51] K. H. Yeap and M. Sarrafzadeh, "A unified approach to floorplan sizing and enumeration", *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1858-1867, Dec. 1993.

[52] P. Gill, W. Murray and M. Suanders, "SNOPT: An SQP algorithm for large scale constrained optimization", *SIAM Journal on Optimization*, vol. 12, pp.979-1006, 2002.

[53] A. E. Dunlop and B. W. Kernighan, "A procedure for placement of standard-cell VLSI circuits", *IEEE Trans. Computer-Aided Design*, vol. 4, pp. 92-98, Jan. 1985.

[54] Y. Zhao and A. Doboli, "Finding broad-scale patterns in large size electronic circuit netlists", *IEEE Midwest Symposium on Circuits and Systems*, Aug. 2005.

[55] A. Gil, K. Passino, S. Gananpathy, "cooperative Task Scheduling for Networked Uninhabited Air Vehicles", *IEEE Transactions on Aerospace and Electronic Systems*, 2007.

[56] J. How, E. King, Y. Kuwata, "Flight Demonstrations of Cooperative Control for UAV Teams", *AIAA "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004.

[57] I. Kaminer, O. Yakimenko, "Cooperative Control of small UAVs for Naval Applications", *43rd IEEE Conference on Decision and Control*, 2004.

[58] D. Rathbun, B. Capozzi, "Evolutionary Approaches to Path Planning Through Uncertain Environments", *Proc. of the american Institute of Aeronautics and Astronautics (AIAA)*, 2002.

[59] D. Shim, H. Chung, H. Kim, S. Sastry, "Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles", *Proceedings of the AIAA GN & C Conference*, 2005.

[60] J. Borenstein, Y. Koren, "Real-Time Obstacle Avoidance for Fast Mobile Robots", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, pp. 1179-1187, Sept/Oct. 1989.

[61] B. Capozzi, J. Vagners, "Evolving (Semi) Autonomous Vehicles", *Proc. of the AIAA Guidance, Navigation and Control Conference*, 2001.

[62] D. Fogel, L. Fogel, "Optimal Routing of Multiple Autonomous Underwater Vehicles through Evolutionary Programming", *Proc. of the Symposium on autonomous Underwater Vehicle Technology*, pp. 44-47, 1990.

[63] B. J. Moore, K. Passino, "Decentralized Redistribution for Cooperative Patrol", *International Journal on Nonlinear and Robust Control*, 2007.

[64] H. Yamaguchi, "A Distributed Motion Coordination Strategy for Multiple Nonholonomic Mobile robots in Cooperative Hunting Operations", *Robotics and Autonomous Systems*, vol. 42, pp. 257-282, Elsevier, 2003.

[65] H. Tanner, A. Jadbabaie, G. Pappas, "Stable Flocking of Mobile Agents, Part I: Fixed Topology", *Proceedings of the IEEE Conference on Decision and Control*, pp. 2010-2015, 2003.

[66] H. Tanner, A. Jadbabaie, G. Pappas, "Stable Flocking of Mobile Agents, Part II: Dynamic Topology", *Proceedings of the IEEE Conference on Decision and Control*, pp. 2016-2021, 2003.

[67] T. Schouwenaars, M. Valenti, E. Feron, J. How, "Linear Programming and Language Processing for Human/Unmanned-Aerial-Vehicle Team Missions", *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 2, March-April, pp. 303-313, 2006.

[68] D. Brogan, J. Hodgins, "Group Behaviors for Systems with Significant Dynamics", *Autonomous Robots*, pp. 135-153, Kluwer, 1997.

[69] O. Brock, O. Khatib, "Real-Time Obstacle Avoidance and Motion Coordination in a Multi-Robot Workcell", *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, pp. 274-279, 1999.

[70] N. Leonard, E. Fiorelli, "Virtual Leaders, Artificial Potentials and Coordinated Control of Groups", *Proceedings of the IEEE Conference on Decision and Control*, pp. 2968-2973, 2001.

[71] M. Mamei, F. Zambonelli, L. Leonardi, "Distributed Motion Coordination with Co-Fields: a Case Study in Urban Traffic Management", *Proc. International Symposium on Autonomous Decentralized Systems*, 2003.

[72] H. Van Dyke Parunak, S. Brueckner, J. Sauter, "Digital Pheromone Mechanisms for Coordination of Unmanned Vehicles", *AAMAS*, 2002.

[73] J. Mitchell, D. Keirsey, "Planning Strategic Paths through Variable Terrain Data", *Proc. of the SPIE Conference on Appplications of Artificial Intelligence*, vol. 4, pp. 172-179, 1984.

[74] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments", *Proc. of the International Conference on robotics and Automation*, vol. 4, pp. 3310-3317, 1994.

[75] O. Yakimenko, V. Dobrokhodov, "Airplane trajectory control at the stage of rendezvous with maneuvering object algorithms synthesis", *IEEE*, 1998.

[76] O. Yakimenko, "Direct method for rapid prototyping of near-optimal aircraft trajectories", *AIAA Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, pp. 865-875, 2000.

[77] A. Goossens, G. Koeners, J. Tadema, E. Theunissen, "Using Simulation to Refine UAV Operator Station Functional Requirements", *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004.

[78] J. Bellingham, "Coordination and Control of UAV Fleets using Mixed-Integer Linear Programming", *Master of Science Thesis*, MIT, 2002.

[79] D. Coombs, M. Herman, T. Hong, M. Nashman, "Real-time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow", *ICCV*, pp. 276-283, 1995.

[80] J. Cortes, S. Martinez, T. Karatas, F. Bullo, "Coverage Control for Mobile Sensing Networks", *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 1327-1332, 2002.

[81] P. Fahlstrom and T. Gleason, *Introduction to UAV Systems*, UAV Systems INC., second edition, 1998.

[82] V. Gazi, K. Passino, "Stability Analysis of Swarms", *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 692-697, April 2003.

[83] V. Gazi, K. Passino, "A Class of Attractions/Repulsion Functions for Stable Swarm Aggregations", *International Journal of Control*, vol. 77, no. 18, pp. 1567-1579, 2004.

[84] M. Gendreau, A. Hertz, G. Laporte, "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science*, vol. 40, no. 20, pp. 1276-1290, October 1994.

[85] R. Hooke and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems", *Journal of the Association for Computing Machinery*, vol. 8, no. 2, pp. 212-229, 1961.

[86] H. Kim, D. Shim, S. Sastry, "Flying Robots: Sensing, Control and Decision Making", *International Conference on Robotics and Control*, 2002.

[87] Y. Liu, M. Simaan, J. Cruz, "an Application of Dynamic Nash Task Assignment Strategies to Multi-Team Military Air Operations", *Automatica*, vol. 39, pp. 1469-1478, 2003.

Appendix A

MCNC Benchmark Floorplanning Graphs

The best floorplanning solutions of MCNC benchmark examples (ami49, ami33, hp, xerox, and apte) obtained by Hercules are given in Fig. A.1 – Fig. A.5 for references.

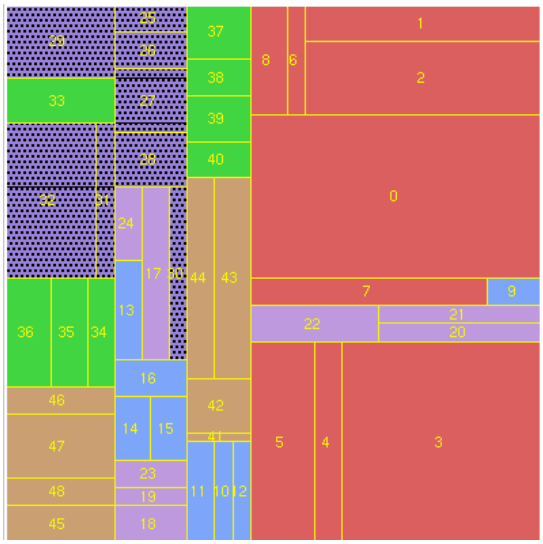

Figure A.1: Final optimum floorplan obtained by Hercules for ami49

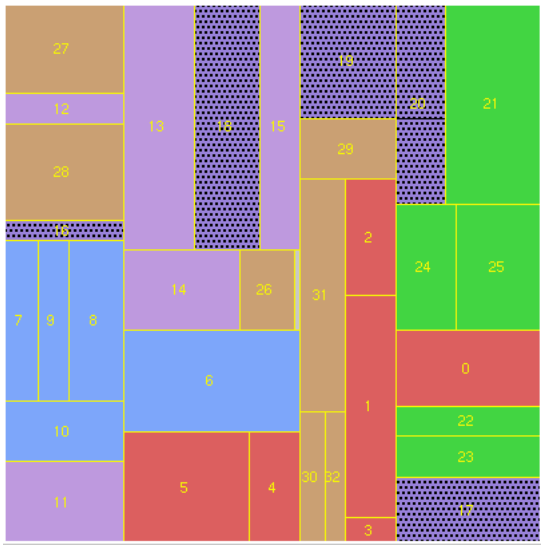

Figure A.2: Final optimum floorplan obtained by Hercules for ami33

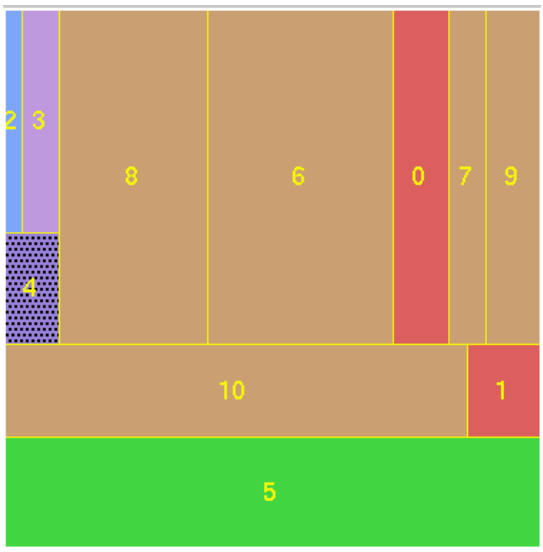

Figure A.3: Final optimum floorplan obtained by Hercules for hp

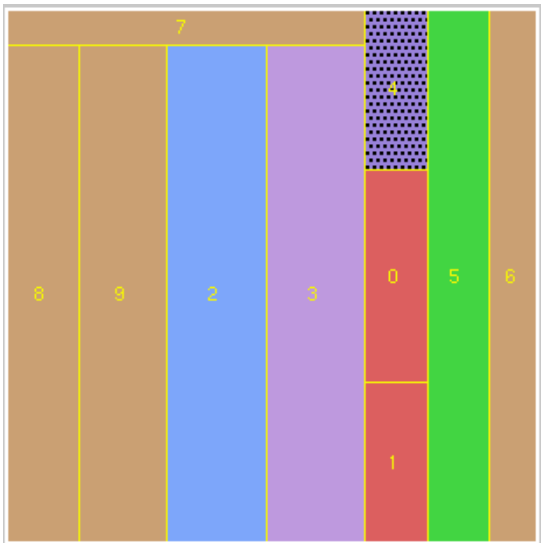

Figure A.4: Final optimum floorplan obtained by Hercules for xerox

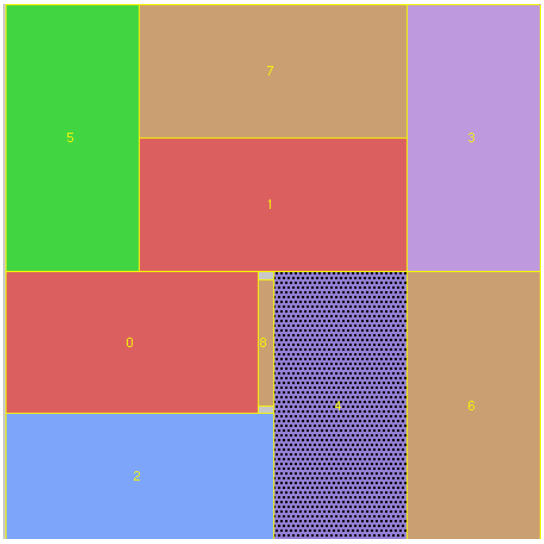

Figure A.5: Final optimum floorplan obtained by Hercules for apte

Appendix B

Collaborating UAVs' Constraints for 3-Thread

1. **Scheduling tasks in the same thread for thread 1 only**

$$T1 \leq T2\_start; \tag{B.1}$$

$$T2\_end = T2\_start + TF1 * x2\_1 + TF1 * x2\_2; \tag{B.2}$$

$$x2\_1 + x2\_2 = 1.0; \tag{B.3}$$

$$T2 \leq T3\_start; \tag{B.4}$$

$$T3\_end = T3\_start + 10 * x3\_1 + 10 * x3\_2; \tag{B.5}$$

$$x3\_1 + x3\_2 = 1.0; \tag{B.6}$$

$$T3 \leq T4\_start; \tag{B.7}$$

$$T4\_end = T4\_start + 20 * x4\_1 + 20 * x4\_2; \tag{B.8}$$

$$x4\_1 + x4\_2 = 1.0; \tag{B.9}$$

Same scheduling tasks' equations for thread 2 and thread 3.

2. **Constraints for task T14**

$$T4\_end \leq T14; \tag{B.10}$$

$$T7\_end \leq T14; \tag{B.11}$$

$$T10\_end \leq T14; \tag{B.12}$$

3. **Constraints for tasks' scheduling for successive threads**

$$1M = 1000000; \tag{B.13}$$

Thread scheduling for thread 1 & 2.

$$T4\_end \leq T5\_start + 1M * z2\_1 + 2M - 1M * x4\_1 - 1M * x5\_1; \tag{B.14}$$

$$T4\_end \leq T5\_start + 1M * z2\_1 + 2M - 1M * x4\_2 - 1M * x5\_2; \tag{B.15}$$

$$T7\_end \leq T2\_start + 1M * z1\_2 + 2M - 1M * x7\_1 - 1M * x2\_1; \tag{B.16}$$

$$T7\_end \leq T2\_start + 1M * z1\_2 + 2M - 1M * x7\_2 - 1M * x2\_2; \tag{B.17}$$

Thread scheduling for thread 1 & 3.

$$T4\_end \leq T8\_start + 1M * z3\_1 + 2M - 1M * x4\_1 - 1M * x8\_1; \tag{B.18}$$

$$T4\_end \leq T8\_start + 1M * z3\_1 + 2M - 1M * x4\_2 - 1M * x8\_2; \tag{B.19}$$

$$T10\_end \leq T2\_start + 1M * z1\_3 + 2M - 1M * x10\_1 - 1M * x2\_1; \tag{B.20}$$

$$T10\_end \leq T2\_start + 1M * z1\_3 + 2M - 1M * x10\_2 - 1M * x2\_2; \tag{B.21}$$

Thread scheduling for thread 2 & 3.

$$T7\_end \leq T8\_start + 1M * z3\_2 + 2M - 1M * x7\_1 - 1M * x8\_1; \tag{B.22}$$

$$T7\_end \leq T8\_start + 1M * z3\_2 + 2M - 1M * x7\_2 - 1M * x8\_2; \tag{B.23}$$

$$T10\_end \leq T5\_start + 1M * z2\_3 + 2M - 1M * x10\_1 - 1M * x5\_1; \tag{B.24}$$

$$T10\_end \leq T5\_start + 1M * z2\_3 + 2M - 1M * x10\_2 - 1M * x5\_2; \tag{B.25}$$