Reliable Deployment of Component-based Applications into Distributed Environments

Abbas Heydarnoori Farhad Mavaddat
University of Waterloo,
Waterloo, ON,
Canada, N2L 3G1

Abstract

Software deployment process is a sequence of related activities for installing an already developed application into its target environment, and bringing it into an executing state. For complex component-based applications that should be deployed into a large distributed environment, several deployment configurations are typically possible. These deployment configurations can have significant impacts on the application's quality of service properties such as reliability. In distributed systems, the reliability of the application is highly dependent on the reliability of its network, and network failures can have adverse effects on the application's reliability. Thus, one possible way to increase the reliability of a distributed component-based application is to deploy it so that the communications among its components are done as local as possible. In this paper, a graph-based deployment planning approach is proposed for this purpose.

keywords: *Software deployment, Software reliability, Component-based applications, Distributed systems, Communication channels.*

1 Introduction

With significant advances in software development technologies, it is possible to have complex software systems that consist of a large number of heterogeneous components distributed over many hosts with different hardware and software characteristics. However, in these applications, different components of the application may have various hardware and software requirements, and hence they may provide their desired functionality only when these requirements are answered. Consequently, after the development of an application, a sequence of related activities must be done to place its components into the suitable hosts in the distributed environment, and to make the application available for use. This sequence of activities is referred to as *software deployment process*, and comprises the following activities: acquiring the developed application from its producer; planning where and how different components of the application should be installed in the target environment, resulting in a deployment plan; installing the application into its target environment according to its deployment plan; configuring it; and finally executing it.

For complex applications deployments that many components should be deployed into many hosts in a distributed environment, several deployment configurations are usually possible. In a given context, some of these deployment configurations are obviously more effective than others in terms of some quality of service (QoS) characteristics such as reliability, performance, dependability, and so on. The main aim of the present work is to maximize the reliability, defined as the probability of failure-free software operation for a specified period of time in a specified environment [1]. In the context of distributed environments, one potential problem is network failures. In these environments, connectivity losses can lead to disastrous effects on the system's reliability, and the software application may not provide its desired functionality. To reduce the risks of this problem, one solution is to make the communications among the application components as local as possible. In this way, components located in the same host can communicate without any respect to the network's status. Thus, from this perspective, the most reliable deployment configuration can be defined as one with the least amount of communications among the hosts in the distributed environment. From another point of view, this can be seen as the increased performance. This is due to the fact that in distributed environments, network communications have some overheads on the software application. Thus, reduced communication among hosts can result in a higher performance.

In this paper, a graph-based approach is presented for planning the deployment of channel-based component-based applications into peer-to-peer distributed environments (e.g., Internet) so that the reliability of the application is maximized. A channel is a peer-to-peer communication medium with well-defined characteristics and behavior [2]. Examples of channel-based models are Reo [2], IWIM [3], and Manifold [4]. In a peer-to-peer architecture, two or more computers (called nodes) can directly communicate with each other, without requiring any intermediary devices [5]. In contrast to the client/server architecture, in a peer-to-peer architecture, nodes have equivalent responsibilities, enabling applications that focus on collaboration and communication. Therefore, the deployment planning approach presented in this paper does the planning with respect to the various channel types (or implementations) required by different components of the application, and various channel types (or implementations) that different hosts in the target environment can support.

This paper is organized as follows. Section 2 talks about the user-specified inputs that are used in the reliable deployment planning. Then, Section 3 describes how these inputs are used to do the actual reliable deployment planning with the help of graph theory algorithms. Finally, Section 4 presents concluding remarks and outlines future work.

2 Deployment Planning Inputs

To generate deployment plans, the following inputs should be specified by the user: (1) the software application being deployed, (2) the distributed environment in which the application will be deployed, and (3) the user-defined constraints regarding this deployment. In the following sections, we talk about these inputs.

2.1 Specification of the Software Application being Deployed

This input specifies the software application being deployed into the target environment. In the view of this paper, a software application comprises a number of software components connected by a number of channels with different characteristics. The nature of these software components are irrelevant to this specification; they are treated as black box software entities that read data from their input ports and write data to their output ports. Thus, they can be processes, Web services, Java beans, CORBA components, and so on. The communications among these black box software entities is done via channels among them. These channels can have different types or implementations. For example, channel types can be synchronous, asynchronous, or FIFO. Examples of channel implementations are simple message passing, encrypted message passing, or using the shared memory. It is also possible to model the primitives of other communication models (such as message passing, shared spaces, or remote procedure calls) by the channel-based communication model [2]. Thus, other kinds of component-based applications can also be seen as some sorts of channel-based component-based applications. With respect to this discussion, a component-based application can be modeled as a graph. The nodes of this graph represent the components of the application. The edges of this graph represent the channels among the application components.

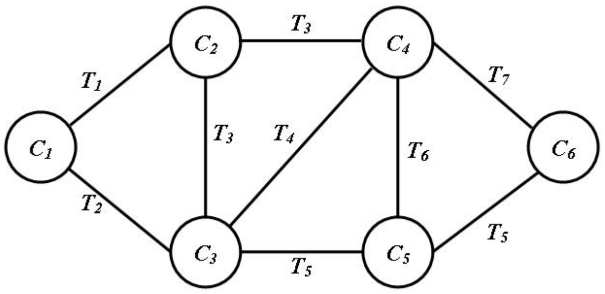


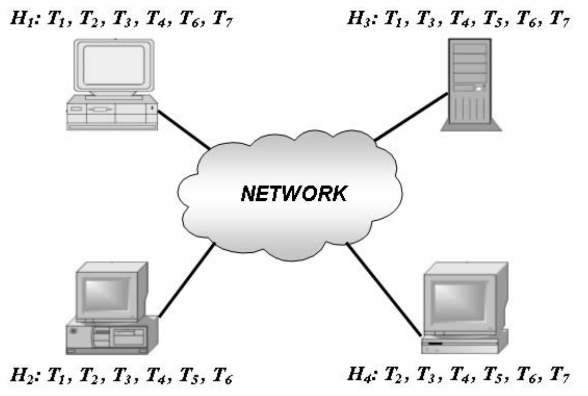
Figure 1. A sample application graph. $\mathrm{C_i}$s represent application components. $\mathrm{T_d}$s represent different channel types among them.

Definition 2.1 ***(Application Graph)*** *Suppose C_is represent different components of the application, and T_ds represent different channel types. Then, application graph* $\mathrm{AG} = (V_{\mathrm{AG}}, E_{\mathrm{AG}})$ *is defined as a graph on* $V_{\mathrm{AG}} = \{C_1, C_2, ..., C_n\}$ *in which each edge* $e \in E_{\mathrm{AG}}$ *has a label* $l_e \in \{T_1, T_2, ..., T_k\}$.

For example, Fig. 1 shows a sample application graph for a supposed software system consisting of six components $C_1 - C_6$ connected by different channel types $T_1 - T_7$.

2.2 Specification of the Target Environment

In this paper, the target environment for the deployment of the application is a peer-to-peer distributed en-



H1: T1, T2, T3, T4, T6, T7
H3: T1, T3, T4, T5, T6, T7
NETWORK
H2: T1, T2, T3, T4, T5, T6
H4: T2, T3, T4, T5, T6, T7

Figure 2. A sample distributed environment consisting of four hosts $\mathbf{H_1 - H_4}$ supporting different channel types $\mathbf{T_1 - T_7}$.

vironment consisting of a number of hosts with computational capabilities (e.g., PCs, laptops, servers, etc.). Furthermore, the required software for the communication among the application components has been already installed on them. However, since different hosts may have different hardware properties, they may not be able to support some features of the communication software installed on them, or it may be even impossible to install some sorts of communication software on them. It is also possible that different features/versions of the communication software have been installed on different hosts intentionally because of some reasons (e.g., cost, security, etc.). With respect to this discussion, available hosts in the target environment might be able to support different sets of channel types (or implementations). As an example, Fig. 2 demonstrates a sample peer-to-peer distributed environment consisting of four hosts $H_1 - H_4$, each of them can support different subsets of channel types $T_1 - T_7$. To create a channel of type T_d between two not necessarily distinct hosts H_x and H_y in this environment, both of them must be able to support that channel type T_d.

2.3 Specification of the User-defined Constraints

Users may have special constraints regarding the placement of the application components that should be taken into account during the deployment planning. For example, suppose users want the deployment of the application presented in Fig. 1 into the target environment presented in Fig. 2 to be done so that component C_5 to be run on either H_1 or H_2.

Component Name	Candidate Hosts
C_1	H_1, H_2
C_2	H_1, H_2, H_3
C_3	H_2, H_4
C_4	H_1, H_3, H_4
C_5	H_2
C_6	H_3, H_4

Table 1. Candidate hosts for the deployment of the application components presented in Fig. 1 into the target environment presented in Fig. 2.

3 Deployment Planning

In this section, the inputs specified in Section 2 are used to do the actual deployment planning. However, first we introduce the concept of candidate host which is used in the rest of this paper.

Definition 3.1 ***(Candidate Host)*** *For the given application graph* AG $= (V_{\mathrm{AG}}, E_{\mathrm{AG}})$*, and set of channel types* $\{T_1, T_2, ..., T_k\}$*, suppose* $T_{C_i} = \{T_d | T_d \in T, \exists \{C_i, C_j\} \in E_{\mathrm{AG}} : l_{\{C_i,C_j\}} = T_d\}$ *illustrates all channel types connected to the component* C_i *in the application graph* AG*, and* T_{H_x} *shows the set of channel types that host* H_x *can support. Then, host* H_x *is a candidate host for the deployment of component* C_i*, only if (1)* $T_{C_i} \subseteq T_{H_x}$*, and (2) host* H_x *satisfies user-defined constraints regarding the deployment of component* C_i*.*

For example, Table 1 shows the candidate hosts for deploying the components of the application presented in Fig. 1 to the target environment presented in Fig. 2. For a more specific example, consider component C_5. In the application graph presented in Fig. 1, C_5 is connected to the channel types T_5 and T_6. By looking at the channel types that each of the hosts in the target environment can support, we see that hosts H_2, H_3 and H_4 can support component C_5's required channel types. But, as mentioned in section 2.3, users want C_5 to be deployed on either hosts H_1 or H_2. Thus, with respect to this constraint, the only candidate host for the deployment of component C_5 is H_2.

One impractical way for finding the most reliable deployment configuration is to generate all possible deployment configurations by permuting the sets of candidate hosts for different components of the application. Then, the deployment configuration with the greatest number of local channels among the application components (or the least number of channels among the hosts) is selected. However, when the number of possible deployment configurations is large, a set of algorithms and heuristics should be designed and applied to effectively solve this problem. Following section presents how this can be done in polynomial time.

3.1 Using Multiway Cut Problem in Reliable Deployment Planning

In this section, we want to find an efficient algorithm for solving this complex problem in polynomial time. For this purpose, we show that the reliable deployment problem corresponds to the multiway cut problem in graph theory [6].

Definition 3.2 ***(Multiway Cut Problem)*** *Let $G = (V, E)$ be an undirected graph on $V = \{v_1, v_2, ..., v_n\}$ in which each edge $e \in E$ has a non-negative weight $w(e)$, and let $T = \{t_1, t_2, ..., t_m\} \subseteq V$ be a set of terminals. Multiway cut is the problem of finding a set of edges $E' \subseteq E$ such that the removal of E' from E disconnects each terminal from all other terminals, and solution cost* $\mathrm{MC} = \sum_{e \in E'} w(e)$ *is also minimized.*

Suppose $\mathrm{AG} = (V_{\mathrm{AG}}, E_{\mathrm{AG}})$ is the application graph of the software application being deployed, $V_{\mathrm{TG}} = \{H_1, H_2, ..., H_m\}$ represents the set of available hosts in the target environment, and CH_{C_i} represents the set of candidate hosts for the deployment of component C_i. To solve the reliable deployment problem, a graph $G = (V, E)$ is made in the following way:

- $V = V_{\mathrm{AG}} \cup V_{\mathrm{TG}}$.
- $E = E_{\mathrm{AG}} \cup E_H$, where $E_H = \{\{C_i, H_x\} | C_i \in V_{\mathrm{AG}}, H_x \in \mathrm{CH}_{C_i}\}$.
- $w(e) = \begin{cases} 1 & e \in E_{\mathrm{AG}} \\ n^2 & e \in E_H \end{cases}$. Here n^2 shows a large number.

Fig. 3 shows an example of a graph developed in this way for the application graph presented in Fig. 1, and the target environment presented in Fig. 2. In this graph, if we set hosts as the terminals of the multiway cut problem, we prove in the following theorem that the solution of the multiway cut problem is the solution of the reliable deployment problem we intend to solve.

Theorem 3.1 *Suppose graph $G = (V, E)$ is built in the way mentioned earlier, and hosts of the target environment are set as the terminals. Then, the multiway cut solution of this graph is the solution of the reliable deployment problem we are looking for. This means that the application components that lie in the same subgraph with a host should be deployed on that host, and this deployment configuration has the least number of channels among hosts.*

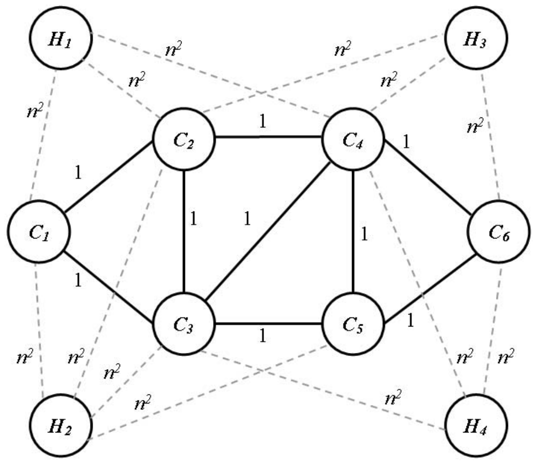


Figure 3. A graph built for finding the most reliable deployment configuration of the application presented in Fig. 1 into the target environment shown in Fig. 2.

Proof Suppose n represents the number of components of the application, and k represents the size of the E_H, i.e., $k = |E_H|$. In the multiway cut solution we are looking for, each component must be assigned to exactly one host. Thus, $(k - n)$ edges whose total weight is $n^2(k - n)$ will be removed from the E_H in the cut. Also, suppose L_{OPT} represents the solution of the reliable deployment problem, i.e., the least number of channels among hosts after the deployment of the application. Actually, these channels are those application graph edges that lie in the cut, and their total weight is $L_{\mathrm{OPT}} \times 1 = L_{\mathrm{OPT}}$. Thus, our goal is to prove that $\mathrm{MC} = n^2(k - n) + L_{\mathrm{OPT}}$.

Case A: $\mathbf{MC \leq n^2(k - n) + L_{OPT}}$.

Suppose a deployment $D : V_{\mathrm{AG}} \longrightarrow V_{\mathrm{TG}}$ whose cost is optimum is done, i.e., it has L_{OPT} number of channels among hosts. Now, assume that $\mathcal{C}$ is its corresponding cut in the graph $G = (V, E)$:

$$\mathcal{C} = \underbrace{\{\{C_i, H_x\} | D(C_i) \neq H_x, \{C_i, H_x\} \in E_H\}}_{\mathcal{M}} \bigcup \underbrace{\{\{C_i, C_j\} | D(C_i) \neq D(C_j), \{C_i, C_j\} \in E_{\mathrm{AG}}\}}_{\mathcal{N}}$$

$\mathcal{M}$ represents the set of edges of E_H that lie in the cut, and $\mathcal{N}$ represents the set of edges of E_{AG} that lie in the cut. The size of $\mathcal{M}$ is $(k - n)$ and the size of $\mathcal{N}$ is L_{OPT}. Furthermore, the weight of the edges in $\mathcal{M}$ is n^2 and the weight of the edges in $\mathcal{N}$ is 1. With respect to this description:

$w(\mathcal{C}) =$
$w(\{\{C_i, H_x\}|D(C_i) \neq H_x, \{C_i, H_x\} \in E_H\})+$
$w(\{\{C_i, C_j\}|D(C_i) \neq D(C_j), \{C_i, C_j\} \in E_{\text{AG}}\}) =$
$n^2 \times |\{\{C_i, H_x\}|D(C_i) \neq H_x, \{C_i, H_x\} \in E_H\}|+$
$1 \times |\{\{C_i, C_j\}|D(C_i) \neq D(C_j), \{C_i, C_j\} \in E_{\text{AG}}\}| =$
$n^2(k-n) + L_{\text{OPT}}$

Since MC is the cost of the optimum multiway cut, for sure, MC $\leq w(\mathcal{C})$. Therefore, MC $\leq n^2(k-n) + L_{\text{OPT}}$.

Case B: $\mathbf{MC \geq n^2(k-n) + L_{OPT}}$.

Suppose $\mathcal{C}$ is the optimum multiway cut for graph $G = (V, E)$ whose cost is MC. Now, we want to use this cut to generate its corresponding deployment D. For this purpose, we prove the following subcases:

Subcase B.1: Cut $\mathcal{C}$ includes at most $(k-n)$ edges of E_H.

Suppose we want to find a cut whose cost is the heaviest. In the deployment configuration we are looking for, each component should be assigned to exactly one host. For this purpose, for each component C_i in graph G, we keep an arbitrary edge connecting that component to an arbitrary host, and we cut the rest of the edges in E_H and E_{AG}. Since the maximum number of edges in the application graph is $\binom{n}{2}$, the cost of this cut is at most $\binom{n}{2} + n^2(k-n)$. Thus, the cost of the multiway cut $\mathcal{C}$ can not be more than $\frac{n^2}{2} + n^2(k-n)$. This means that the cut $\mathcal{C}$ includes at most $(k-n)$ edges of E_H. Because, for example, if it includes $(k-n+1)$ edges of E_H, then the cost of the cut would be $\binom{n}{2} + n^2(k-n+1)$ which is more than the maximum cost we found here.

Subcase B.2: Each component C_i is connected to at most one host in the cut $\mathcal{C}$.

Suppose a component C_i is connected to two different hosts H_x and H_y in the cut. This means that H_x and H_y are connected together in the cut. However, since H_x and H_y belong to the set of terminals, this is impossible. Therefore, C_i is connected to at most one host in the cut.

Subcase B.3: Each component C_i is connected to exactly one host in the cut $\mathcal{C}$.

From subcases B.1 and B.2 together, it can be easily understood that each component C_i is connected to exactly one host in the cut $\mathcal{C}$. $D(C_i)$ represents the host on which component C_i is mapped.

By using the subcase B.3, cut $\mathcal{C}$'s corresponding deployment configuration D can be made. Suppose $L_D = |\{\{C_i, C_j\}|D(C_i) \neq D(C_j), \{C_i, C_j\} \in E_{\text{AG}}\}|$ represents the cost of the deployment configuration D, i.e., the number of channels among the hosts in the deployment configuration D. In the following, we prove the correctness of case B:

$$\begin{array}{lcl} \text{MC} & = & n^2(k-n)+ \\ & & |\{\{C_i, C_j\}|\{C_i, C_j\} \in \mathcal{C}, \{C_i, C_j\} \in E_{\text{AG}}\}| \\ & \geq & n^2(k-n)+ \\ & & |\{\{C_i, C_j\}|D(C_i) \neq D(C_j), \{C_i, C_j\} \in E_{\text{AG}}\}| \\ & & = n^2(k-n) + L_D \\ \Longrightarrow & & \text{MC} \geq n^2(k-n) + L_D \geq n^2(k-n) + L_{\text{OPT}} \end{array}$$

Cases A and B together imply that MC $= n^2(k-n) + L_{\text{OPT}}$. Therefore, the correctness of theorem 3.1 is proved. ∎

```
1. For each terminal $t_i \in T$, find a minimum-cost
   set of edges $\mathcal{C}_{t_i}$ whose removal disconnects $t_i$
   from the rest of the terminals;
2. Discard cut $\mathcal{C}_{t_x}$ whose cost $w(\mathcal{C}_{t_x})$ is the
   heaviest;
3. Output the union of the rest, call it $\mathcal{C}$.
```

Algorithm 1: Approximation algorithm for solving the multiway cut problem.

In theorem 3.1, we showed that the solution of the reliable deployment problem can be found by solving the multiway cut problem in graph theory. However, it is proved that the multiway cut problem is an NP-hard problem when the number of terminals is greater than two. Thus, unless P=NP, it does not have a polynomial time solution [6]. However, it is possible to find many approximation algorithms for the multiway cut problem in literature [6, 7, 8]. One of the well-known and simple approximation algorithms developed by Dalhaus et al. is provided in Algorithm 1 [6]. As an example, Fig. 4 shows an example of applying this algorithm on the graph presented in Fig. 3. As we see in this figure, one of the main problems of these approximation algorithms is that some components may not be assigned to any hosts (e.g., C_4 and C_6). To solve this problem, after applying the multiway cut approximation algorithm on the graph, we check whether or not all components are assigned to a host. If there are

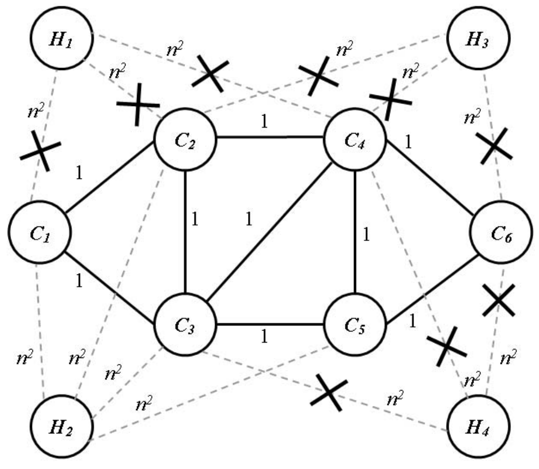


Figure 4. An approximation for the multiway cut of the graph presented in Fig. 3.

some components which are not assigned to any hosts, we connect those components to one of their candidate hosts for the deployment, and we cut all the application graph edges that are connected to those components. Since, we are actually removing from the multiway cut approximation some heavy edges that connect the components to the hosts, this approach not only will solve the problem, but also will improve the approximation of the multiway cut. Consequently, the result is closer to the optimum solution we are looking for. After applying this improvement on the multiway cut approximation presented in Fig. 4, one possible solution for the reliable deployment problem is $\{(C_1 \mapsto H_2), (C_2 \mapsto H_2), (C_3 \mapsto H_2), (C_4 \mapsto H_4), (C_5 \mapsto H_2), (C_6 \mapsto H_4)\}$.

4 Conclusions and Future Work

This paper presented a graph-based approach for maximizing the reliability of component-based applications deployments into distributed environments. The reliability of a distributed application is extremely dependent on the reliability of its network. Therefore, in the deployment planning approach presented in this paper, the communications among the application components were tried to be made as local as possible, allowing minimization of the network failures effects on the application's reliability. Furthermore, it was demonstrated that this deployment problem corresponds to the multiway cut problem in graph theory that is a NP-hard problem. In this paper, an existing multiway cut approximation algorithm with some improvements was used to solve the reliable deployment problem. For future work, we plan to develop a number of heuristics that can provide a better approximation of the reliable deployment problem.

Acknowledgment

The authors would like to acknowledge Ehsan Chiniforooshan for his valuable suggestions in developing the ideas expressed in this paper.

References

[1] Lyu, M. R. *Handbook of Software Reliability Engineering*, IEEE Computer Society Press and McGraw-Hill, 1996.

[2] Arbab, F. Reo: A Channel-based Coordination Model for Component Composition. *Mathematical Structures in Computer Science*, 14, 3 (June 2004), 329-366.

[3] Katis, P., Sabadini, N. and Walters, R. F. C. A Formalization of the IWIM Model. In *Proceedings of the 4th International Conference on Coordination Languages and Models (COORDINATION 2000)*, Limassol, Cyprus, September 11-13, 2000.

[4] Bonsangue, M. M., Arbab, F., de Bakker, J. W., Rutten, J., Scutell, A. and Zavattaro, G. A Transition System Semantics for the Control-driven Coordination Language Manifold. *Theoretical Computer Science*, 240, 1 (June 2000), 3-47.

[5] Schollmeier, R. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In *Proceedings of the IEEE 2001 International Conference on Peer-to-Peer Computing (P2P2001)*, Linkping, Sweden, August 27-29, 2001.

[6] Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D. and Yannakakis, M. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23, 4 (August 1994), 864-894. Preliminary version appeared in STOC'92.

[7] Calinescu, G., Karloff, H., and Rabani, Y. An Improved Approximation Algorithm for Multiway Cut. *Journal of Computer and System Sciences*, 60, 3 (June 2000), 564-574. Preliminary version in STOC'98.

[8] Vazirani, V. V. *Approximation Algorithms*, Second Edition, Springer, 2002.