Welcome to SENG 480A / CSC 485A / CSC 586A Self-Adaptive and Self-Managing Systems

Dr. Hausi A. Müller
Department of Computer Science
University of Victoria

http://courses.seng.uvic.ca/courses/2015/summer/seng/480a
http://courses.seng.uvic.ca/courses/2015/summer/csc/485a
http://courses.seng.uvic.ca/courses/2015/summer/csc/586a

1

Announcements

- A4
  - Posted
  - Due Friday, July 31
  - Adaptive control
- Teaching evaluations
- Grad project
  - Due Friday, July 24
  - Presentations Mon, July 27 and Thu, July 30
  - All students are expected to assess presentation as part of course participation mark

2

Teaching Evaluations
CES —Course Evaluation Survey

- Your responses are important to me and TAs
- Your responses are important for future students
- Your responses are important to Department Chair and Dean
- Completing CESs is good university citizenship
- Complete CES at http://ces.uvic.ca
  - Sign in to UVic
  - Conduct survey
  - Can be 'saved' and 'submitted' later
  - Works on desktops or mobile devices
  - Survey closes at end of last day of class
- Survey results available to instructors after grade submission

3

Graduate Student Research Paper Presentations

- Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H. Kienle, H.M., Litoiu, M., Muller, H.M., Pezzè, M., Shaw, M. Engineering Self-Adaptive Systems through Feedback Loops. Software Engineering for Self-Adaptive Systems, pp. 48–70 (2009) — **Presentation by Simar Arora Khushboo Gandhi: July 27**
- Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., Steenkiste, P. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. IEEE Computer 37(10) 46–54 (2004) — **Presentation by Stephan Heinemann and Waseem Ullah: July 27**
- Orezzy, P., Medvidovic, N., Taylor, R.N.. Runtime Software Adaptation: Framework, Approaches, and Styles. In: ACM/IEEE International Conference on Software Engineering (ICSE 2008), pp. 899–910 (2008) — **Presentation by Sumit Kadyan and Adithya Rathakrishnan: July 27**
- Kramer, J., Magee, J.. Self-Managed Systems: An Architectural Challenge. In: ACM /IEEE International Conference on Software Engineering 2007 Future of Software Engineering (ICSE), pp. 259–268 (2007) — **Presentation by Ernest Aaron and Harshit Jain : July 27**

4

Graduate Student Research Paper Presentations

- Aksanli, J. Venkatesh, L.Z., Tajana R.. Utilizing Green Energy Prediction to Schedule Mixed Batch and Service Jobs in Data Centers. In: Proceedings 4th Workshop on Power-Aware Computing and System (HotPower 2011), Article 5 (2011) — **Presentation by Junnan Lu and Francis Harrison: July 30**
- Ibrahimi, S., Villegas, N.M., Muller, H.A., Thomo, A.. SmarterDeals: a context-aware deal recommendation system based on the SmarterContext engine. CASCON 2012: 116–130 (2012) — **Presentation by Carlene Lebeuf and Maria Ferman: July 30**
- Villegas, N.M., Muller, H.A., Tamura, G., Duchien, L., Casallas, R.. A framework for evaluating quality-driven self-adaptive software systems. In: Proc. 6th Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011), pp. 80–89 (2011) — **Presentation by Parminder Kaur and Navpreet Kaur: July 30**
- Villegas, N.M., G. Tamura, H.A. Müller, L. Duchien, and R. Casallas. DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems. in: R. de Lemos, H. Giese, H.A. Müller, and M. Shaw (Eds.). Software Engineering for Self-Adaptive Systems. LNCS 7475. Dagstuhl Seminar 10431. Springer. pp. 265-293 (2013) — **Presentation by Arturo Reyes Lopez and Babak Tootoonchi: July 30**

5

Guidelines for Grad Student Presentations

- Format of presentation
  - Presentation 10 mins
  - Q&A 5 mins
  - Practice talk (!!)
  - Practice of the best of all instructors
- Slides
  - High quality and polished
  - Submit slides by July 24 to instructor for approval
  - Submit final slides 1 day after presentation for posting on website
- Talk outline
  - Motivation
  - Problem
  - Approach
  - Contributions of the paper
  - Relation to what we learned in the course so far
- Assessment
  - All students have to fill out an evaluation form
  - Counts towards class participation

6

Presentation Assessment

Evaluator's name:		
Graduate students:		
Quality of presentation		
Did I learn something? Did the presentation stimulate my interest?	5	
Do I know now what the paper is all about?	5	
Does the presenter know the subject well?	5	
Presentation style: main points reiterated; positive attitude; excited about the subject.	5	
How did the presenter perform in the Q&A session?	5	
Subtotal	**25**	
Other comments		

July 27 and July 30 CSC 586A Presentations

7

Course Requirements

Unit	Undergrads Weight	Grads Weight	Remarks
A1	12%	9%	Due Fri, May 29, 2015
A2	12%	9%	Due Fri, June 19, 2015
A3	12%	9%	Due Fri, July 10, 2015
A4	12%	9%	Due Fri, July 31, 2015
Grad Project		12%	Due Sat, July 25, 2015
Participation and presentation	7%	7%	Only graduate students are required to give a presentation towards the end of the course.
Midterm 1	20%	20%	June 4, 2015 in class. Closed books, closed notes, no phones, no computers, no calculators, no gadgets.
Midterm 2	25%	25%	July 16, 2015 in class. Closed books, closed notes, no phones, no computers, no calculators, no gadgets.
Total	100%	100%	Have a great course!

- All materials discussed in class are required for the midterm examinations
- Completing all midterms and assignments is required to pass the course
- Passing the midterms is not absolutely required to pass the course, but of course highly recommended

8

Assignment 4

Part I

In Part I you are to write a summary of the following paper:

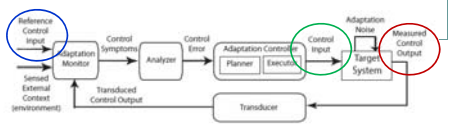
Villegas, Müller, Tamura, Duchien, Casallas: A framework for evaluatingquality-driven self-adaptive software systems, Proc. 6th Int. Symposium on Software Engineering for Adaptive andSelf-Managing Systems (SEAMS 2011), pp. 80-89 (2011)

The answers to this question should fit into approximately 2-3 typeset pages.

Do not copy verbatim from any source. Cite your sources.

9

The Characterization Model

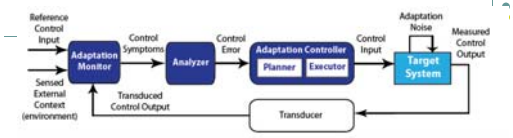


Adaptation goal	Reference inputs	Measured outputs	Computed control actions
Self-* properties, and functional and non-functional requirements	The way how adaptation goals are specified	Values measured in the managed system	The way how the managed system is affected: structural, behavioral
Self-managing	SLAs: average response time per request <= x	Response time per request in an interval of time	1. Assign CPU 2. Process allocation 3. Load balancingb

Villegas, Müller, Tamura, Duchien, Casallas: A framework for evaluating quality-driven self-adaptive software systems, *Proc. 6th Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011)*, pp. 80-89 (2011)

10

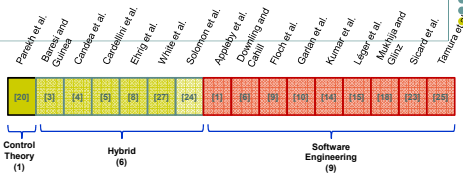
The Characterization Model



System Structure	Adaptation Properties	Evaluation and Metrics
Controller and managed system	Observable and measurable properties for assessing the adaptation	The way how researches are evaluating their approaches
C: MAPE-based MS: Modifiable/ reflection	Settling time	Performance of the adaptation process (response time)

11

The Adaptation Spectrum



Control Actions

Continuous signals affecting behavioral parameters	Discrete operations affecting the software architecture

Managed System's Structure

Non-modifiable structure	Modifiable structure Software models and reflection

12

Catalog of Adaptation Properties

Adaptation Property	Property Verification Mechanism	Where the Property is Observed
Stability	Dynamic	Managed system
Accuracy	Dynamic	Managed system
Settling Time	Dynamic	Both
Small Overshoot	Dynamic	Managed sys
Robustness	Dynamic	Controller
Termination	Static	Controller
Consistency	Both	Managed system
Scalability	Dynamic	Both
Security	Dynamic	Both

From Control Theory

From seminal SAS papers

1. Assign CPU
2. Process allocation
3. Load balancing

13

Adaptation Property	Quality Attributes	
Stability	Performance	Latency Throughput Capacity
	Dependability	Safety Integrity
	Security	Integrity
Accuracy	Performance	Latency Throughput Capacity
Settling Time	Performance	Latency
Small Overshoot	Performance	
Robustness	Dependability	Availability Reliability
	Safety	Interact. Complex. Coupling Strength
Termination	Dependability	Reliability Integrity
Consistency	Dependability	Maintainability Integrity
Scalability	Performance	Latency Throughput Capacity
Security	Security	Confidentiality Integrity Availability

Performance of the adaptation process (response time)

Mapping Properties and QAs

14

Assignment 4

Part II

Control theory offers several reference models for realizing *adaptive control* where not only the target system but also the controller is adjusted over time guaranteeing global stability and convergence. Two famous models are *reference adaptive control (MRAC)* and *model identification adaptive control (MIAC)*. *DYNAMICO* is another reference model for adaptive control.

Villegas, N.M., G. Tamura, H.A. Müller, L. Duchien, and R. Casallas. DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems. in: R. de Lemos, H. Giese, H.A. Müller, and M. Shaw (Eds.). Software Engineering for Self-Adaptive Systems. LNCS 7475. Dagstuhl Seminar 10431. Springer. pp. 265-293 (2013)

In Part II you are to write a tutorial on how to apply the MIAC reference model for designing self-adaptive systems. Explain how MIAC is used using a concrete example. Take this opportunity to immerse yourself in adaptive control.

The answers for this question should fit into approximately 2-3 typeset pages.

Do not copy verbatim from any source. Cite your sources.

15

Adaptive Control

- Adaptive control is the idea of "redesigning" the controller while online, by
  - looking at its performance and
  - changing its dynamic in an automatic way
- Motivated by aircraft autopilot design
  - Allow the system to account for previously unknown dynamics
- Adaptive control uses feedback to observe the process and the performance of the controller and reshapes the controller closed loop behavior autonomously.

16

Adaptive Control

- Modify the control law to cope by changing system parameters while the system is running
- Different from Robust Control in the sense that it does not need *a priori* information about the uncertainties
  - Robust Control includes the bounds of uncertainties in the design of the control law.
  - Therefore, if the system changes are within the bounds, the control law needs no modification

17

Characteristics of Three-Tier Hierarchical Intelligent Control Systems

- The three-tier architecture is prevalent
  - service-oriented software systems
  - automation systems
  - decision-support systems
  - many other types of adaptive and self-managing systems
- Three layers
  - separate concerns (e.g., three-tier web architecture where the presentation and data tiers are separated by an application or business logic tier)
  - Impose a hierarchy along a dimension where such a dimension represents an extra-functional requirement or quality criterion as outlined
    - performance, internal state, goals, policies, plan sophistication, "intelligence", or quality of service
  - The scales depend on the actual requirement or criterion of the dimension
    - from specific goals to general goals
    - from high precision to low precision
    - from fast performance to slow performance
    - from stateless to memory of the past and predictions of the future
    - from hard-wired policies to utility-function policies (i.e., trade-off analysis)
  - Rationale for three tiers is usually not explicitly stated, but frequently a natural fit

18

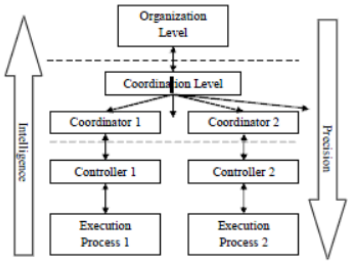
Hierarchical Intelligent Control

- AI and robotics communities generated several closely related three-layer reference control architectures:
  - R. A. Brooks: A Robust Layered Control System for a Mobile Robot, *IEEE Journal on Robotics and Automation* RA-2(1), March 1986.
  - R.J. Firby: *Adaptive Execution in Dynamic Domains,* PhD Thesis, TR YALEU/CSD/RR#672, Yale University, 1989.
  - E. Gat: *Reliable Goal-directed Reactive Control for Real-world Autonomous Mobile Robots,* Ph.D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1991.
  - E. Gat: Three-layer Architectures, Artificial Intelligence and Mobile Robots, MIT/AAAI Press, 1997.
  - T. Shibata & T. Fukuda: Hierarchical Intelligent Control for Robotic Motion, *IEEE Trans. On Neural Networks* 5(5): 823-832, 1994.

19

Hierarchical Intelligent Control System (HICS) Architecture

1986-94



T. Shibata & T. Fukuda: Hierarchical Intelligent Control for Robotic Motion, *IEEE Trans. On Neural Networks* 5(5): 823-832, 1994

20

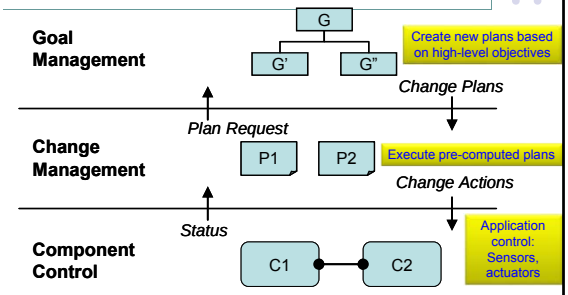
HICS Architecture

- Hierarchical Intelligent Control System (HICS)
- HICS is probably the most general reference architecture emerging from AI and robotics
- Three HICS layers (from bottom to top)
  - Execution
  - Coordination
  - Organization Level
- The complexity of reasoning (i.e., intelligence) increases from the execution to the organization level
- The flexibility of policies decreases from organization to execution (i.e., the precision of increases).

21

Robotics Inspired Three-Layer Architecture Model

2007



Kramer, Magee: Self-Managed Systems—An Architectural Challenge, Future of Software Engineering (FoSE 2007), ICSE 2007.

22

Dimensions of Three-Layer Control System Reference Architectures

Environment uncertainty	Human involvement	Algorithm state	Algorithm specification	Policy flexibility	Goal specificity	Real-time performance	Feedback latency
Significant uncertainty about the environment	Orchestrated in part by humans	Algorithms with state for past memory and future predictions	Deliberative services	Utility-function policies	High level goals and extensive planning	No real-time constraints	Feedback loops with long latency
Medium uncertainty about the environment	Fully autonomic but its policies can be adjusted by humans	Algorithms with state reflecting memory of the past	Task procedures	Goal policies	React and respond to situations using pre-computed plans	Selected real-time constraints	Feedback loops with medium latency
No or minimal uncertainty about the environment	Fully autonomic	Stateless algorithms	Control laws	Action policies	Event and component management	Hard real-time constraints	Feedback loops react quickly

23

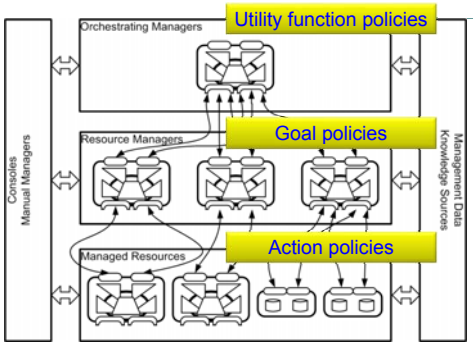
Dimensions of Three-Layer Control System Reference Architectures

ATLANTIS Gat 1991	HICS Shibata & Fukuda 1994	3T Bonasso, Firby, Gat 1997	IBM ACRA 2006	Kramer & Magee 2007	Adaptive SOA 2008
Deliberator	Organization	Planning	Orchestrating managers	Goal management	User management
Sequencer	Coordination	Sequencing	Resource managers	Change management	Workflow management
Controller	Execution	Skill	Managed Resources	Component control	Service management

24

Autonomic Computing Reference Architecture (ACRA)

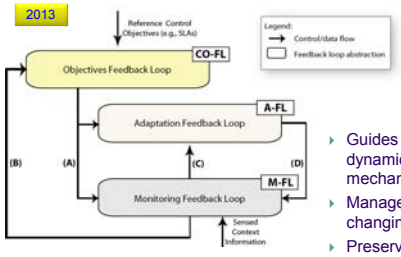
Hierarchy of Autonomic Elements



25

The DYNAMICO Reference Model

2013



Q3: How to maintain context-awareness?

Q4: How to apply dynamic monitoring to runtime V&V?

- Guides the design of highly dynamic self-adaptation mechanisms
- Manages uncertainty due to changing requirements
- Preserves context-awareness in self-adaptation

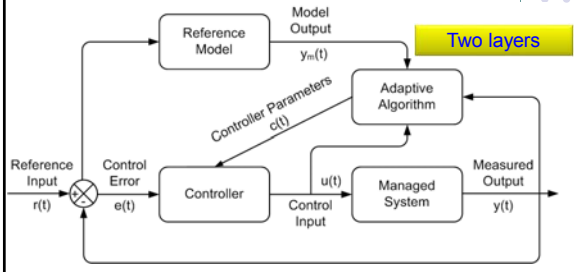
Villegas, Tamura, Müller, *et al.: DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems*, Springer (2013)

28

Adaptive Control—MRAC

Model Reference Adaptive Control

1960



27

Model Reference Adaptive Controllers—MRAC

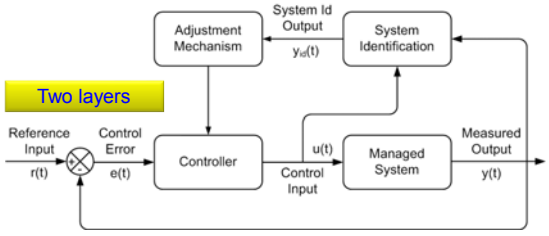
- Also referred to as Model Reference Adaptive System (MRAS)
- Closed loop controller with parameters that can be updated to change the response of the system
- The output of the system is compared to a desired response from a reference model (e.g., simulation model)
- The control parameters are updated based on this error
- The goal is for the parameters to converge to ideal values that cause the managed system response to match the response of the reference model.

28

Adaptive Control—MIAC

Model identification Adaptive Control

1965



Müller and Villegas: Runtime evolution of highly dynamic software, in Evolving Software Systems, Mens, et el. Springer, pp. 229-264 (2014)

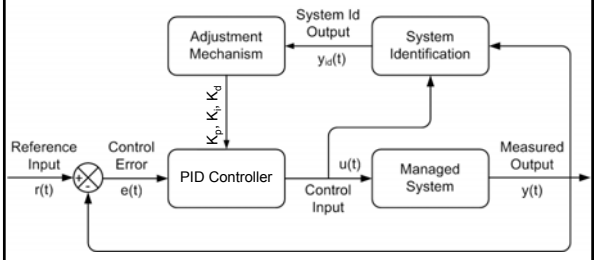
29

Model Identification Adaptive Controllers—MIAC

- Perform system identification while system is running to modify the control laws
  - Create model structure and perform parameter estimation using the Least Squares method
- Cautious adaptive controllers
  - Use current system identification to modify control law, allowing for system identification uncertainty
- Certainty equivalent adaptive controllers
  - Take current system identification to be the true system, assume no uncertainty
  - Nonparametric adaptive controllers
  - Parametric adaptive controllers

30

Model Identification Adaptive Controllers—MIAC



31

System Identification Model Building

- Mathematical tools and algorithms to build dynamical models from measured data
- A dynamical mathematical model in this context is a mathematical description of the dynamic behavior of a system or process in either the time or frequency domain
- Theories and processes
  - Physical
  - Computing
  - Social
  - Engineering
  - Economic
  - Biological
  - Chemical
  - Therapeutic

32

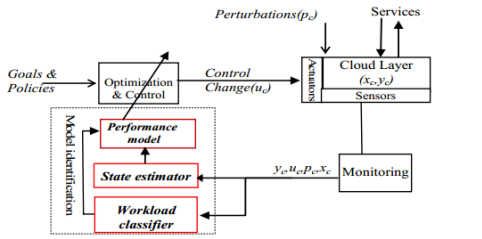
Model Predictive Control (MPC)

- Two-level controllers like controllers for adaptive control
- Model predictive controllers rely on dynamic models of the managed system
- Most often linear empirical models obtained by system identification
- The main advantage of MPC is the fact that it allows the current timeslot to be optimized, while taking future time slots into account
- Optimize a finite time-horizon, but only realize the current timeslot
- MPC has the ability to anticipate future events and can take control actions accordingly
- Generic PID controllers do not have predictive abilities

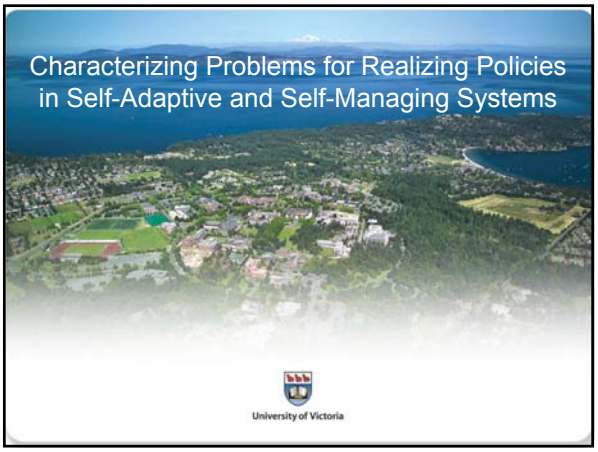
33

Model Predictive Control (MPC)

1990



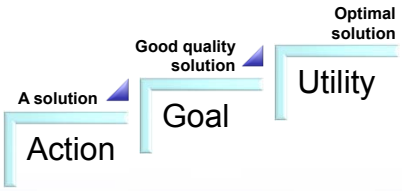
34



Outline

1. Background and related work
2. Characterizing policy-based optimization problems using the *Greedy algorithm*
3. Mathematical framework to add structure to problems to guarantee solution quality
4. Case study — SEAMS studies

Policy framework by Kephart & Walsh

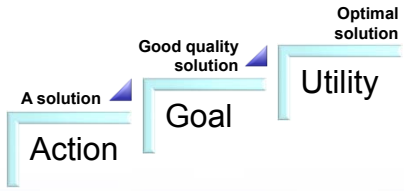


Algorithms increase in sophistication

J. Kephart, W. Walsh: An AI perspective on autonomic computing policies. In: Proc. 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 3-12 (2004)

2

Our approach



Add problem structure for Greedy algorithm

3

Our research question

- Is it possible to add structure to an optimization problem so that the resulting solution—using the Greedy algorithm—can meet requirements of goal and utility function policies?

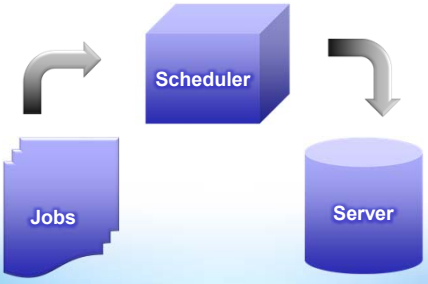
4

Our main contribution

- Is it possible to add structure to an optimization problem so that the resulting solution—using the Greedy algorithm—can meet requirements of goal and utility function policies?
- Yes ➡ using our two mathematical frameworks we can reason about the quality of the resulting solutions

5

A typical SEAMS problem
Data center scheduling



6

Data center scheduling problem

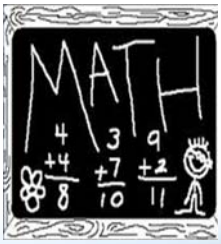
- Given a set of n Jobs J_1, …, J_n each with the following parameters:
  - ❖ Arrival time: A_i
  - ❖ Deadline: D_i
  - ❖ Processing time: P_i
  - ❖ Profit or revenue: R_i

  schedule the jobs on a single server so that the total revenue is maximized.
- The total revenue of a schedule is the sum of the revenues of the jobs processed in the schedule.

7

Our mathematical frameworks

- An optimization problem has two components
  1. Objective function
  2. Set of constraints
- Mathematical frameworks
  1. Objective function based
  2. Constraint based



8