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

Announcements

- A4
  - Posted
  - Due Friday, July 31
  - Adaptive control
- Marks
  - A3 marks posted
  - Refresh if A3 marks are not shown
  - Midterm 2 marks hopefully ready early next week
- Grad project
  - Slides due Friday, July 24
  - Presentations Mon, July 27 and Thu, July 30
  - All students are expected to assess the presentations as part of their course participation mark
- Teaching evaluations
  - Complete CES at http://ces.uvic.ca

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

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**
- Oreizy, 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**

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**
- Ebrahimi, 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**

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

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

Characterizing Problems for Realizing Policies in Self-Adaptive and Self-Managing Systems

University of Victoria

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

Edmond`s Theorem

- **Utility Function Policy:** J. Edmonds in 1971 proved that if an objective function is linear and the constraint set forms a matroid, the greedy algorithm produces an optimal solution.

J. Edmonds: Matroids and the Greedy algorithm. Mathematical Programming Studies, 1(1):27-36 (1971)

Mestre's Theorem

- **Goal Policy:** J. Mestre in 2006 proved that if an objective function is linear and the constraint set forms a k-extendible system, the greedy algorithm gives a 1/k approximation.
- **Approximation Algorithm:** When the quality of solution output by the algorithm is at most factor k away from the optimal solution. This can be thought of as desirable solution.

J. Mestre: Greedy in approximation algorithms. In: Proc. 14th Annual European Symposium on Algorithms (ESA), pp. 528-539 (2006)

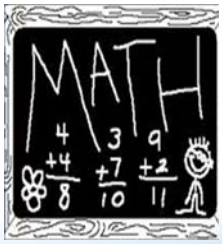
Our main contribution

- 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

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



Handbook for designing policy-driven optimization strategies

Constraints \ Objective function	Linear	Submodular	Unrestricted
Matroid	Optimal Utility Function	½ approximation Goal	No guarantees Action
K-extendible	1/k approximation Goal	1/k+1 approximation Goal	No guarantees Action
Unrestricted	No guarantees Action	No guarantees Action	No guarantees Action

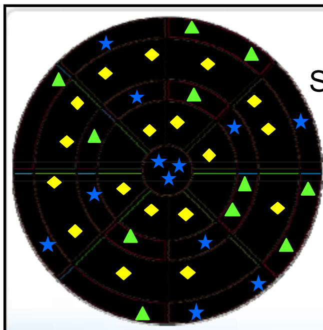
How to use our handbook

- Our characterization and approach helps designers of self-adaptive and self-managing systems:
  - Formulate optimization problems
  - Decide on algorithmic strategies based on policy requirements
  - Reason about solution qualities

Metaphor
Solution quality dartboard

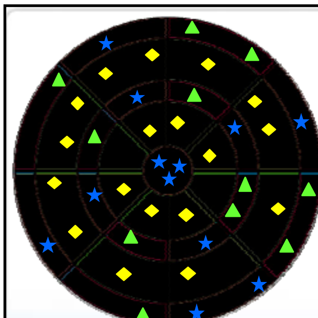
- Regions represent solution qualities
- Aim for high quality regions

Metaphor:
Solution dart board



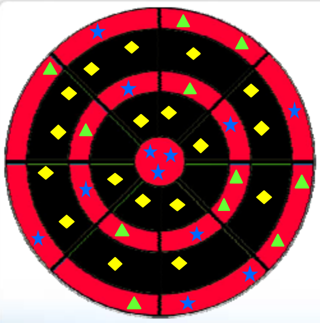
Legend	
★	Optimal solution
▲	Good solution
◆	A solution

Action
dart board



Legend	
★	Optimal solution
▲	Good solution
◆	A solution

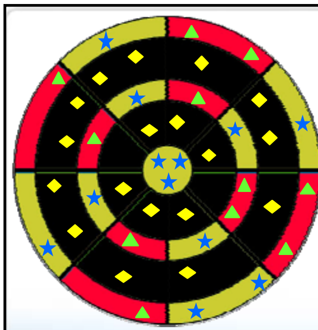
Goal dart board



Legend
- Optimal solution
- Good solution
- A solution

14

Utility function dart board



Legend
- Optimal solution
- Good solution
- A solution

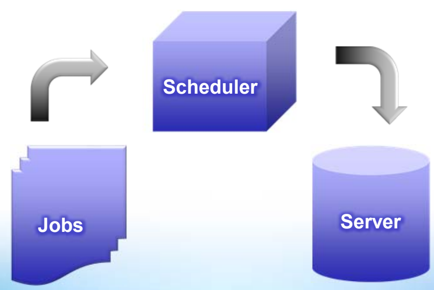
15

SAS applications

- Resource allocation in distributed systems
- Resource allocation in QoS service management
- Data center based scheduling problem
- SLA profit optimization

16

A typical SAS problem Data center scheduling



6

Data center scheduling problem

- Given a set of n Jobs $J_1, \ldots, J_n$ 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

Greedy algorithm

- Sort the jobs based on the revenue R_i
- Start with the empty schedule and add a next job from the sorted list to the current schedule, if feasible

21

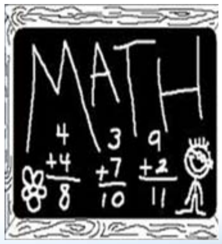
Our mathematical frameworks

- Mathematical frameworks
  1. Objective function based
  2. Constraint based

     Properties

     Downward closure

     Augmentation



8

Linear and Submodular Objective Functions

DEFINITION 2 (SUBMODULAR FUNCTION). *For a given set U, function $g : 2^U \rightarrow \mathbb{R}^+$ is called* submodular *if $g(A \cup B) + g(A \cap B) \leq g(A) + g(B)$ for all $A, B \subseteq U$.*

DEFINITION 3 (LINEAR FUNCTION). *For a given set U, a function $W : 2^U \rightarrow \mathbb{R}^+$ is called* linear *if, for any $F \subseteq U$, $W(F) = \Sigma_{s \in F} w(s)$ for some fixed underlying weight function $w : U \rightarrow \mathbb{R}^+$.*

26

Matroid Constraints

DEFINITION 1 (MATROID [22]). *A set system $(U, \mathcal{F})$, $\mathcal{F} \subseteq 2^U$, is called a matroid if it satisfies the following conditions:*

1. *$\mathcal{F}$ satisfies the* downward-closure *property: If $A \subseteq B$ and $B \in \mathcal{F}$, then $A \in \mathcal{F}$. That is, any subset of a member of the collection $\mathcal{F}$ is also a member of $\mathcal{F}$.*
2. *$\mathcal{F}$ satisfies the* augmentation *property: If $A, B \in \mathcal{F}$ and $|B| > |A|$, then there exists an element x in $B - A$ such that $A \cup x \in \mathcal{F}$. In other words, if we choose two sets A and B from $\mathcal{F}$ such that the size of B is larger than A, then it is possible to move an element x from B to A such that $A \cup x$ also is in $\mathcal{F}$.*

27

K-extensible Constraints

DEFINITION 4. (APPROXIMATION ALGORITHMS [32]). *An algorithm A for a maximization problem P is said to be a ρ-approximation algorithm if for any instance x of P, the value of the objective function on the output of A, denoted by $A(x)$, is at most a factor ρ away from the value of the objective function for the best possible solution, denoted by $OPT(x)$. That is,*

$$\frac{A(x)}{OPT(x)} \geq \rho$$

DEFINITION 5 (k-EXTENDIBLE SYSTEM [20]). *Set system $(U, \mathcal{F})$, $\mathcal{F} \subseteq 2^U$ is called k-extendible if it satisfies the following properties:*

1. *Downward-closure: If $A \subseteq B$ and $B \in \mathcal{F}$, then $A \in \mathcal{F}$.*
2. *Exchange: Let $A, B \in \mathcal{F}$, $A \subsetneq B$ and $x \in U - B$ such that $A \cup \{x\} \in \mathcal{F}$. Then there exists $Y \subseteq B - A$, $|Y| \leq k$ such that $B - Y \cup \{x\} \in \mathcal{F}$. In other words, let us start with any choice of two sets A and B such that B is an extension of A. Suppose that there is an element x such that the set A with x added to it also belongs to $\mathcal{F}$. Then we will be able to find a subset Y inside B of size at most k such that if we remove the elements of Y from B and add the element x to the resulting set, it will also belong to the collection $\mathcal{F}$.*

28

Constraints based framework

- Suppose that the objective function is linear
- Vary the constraint set
- Add structure to the constraint set so that it satisfies the *k-extendibility* or *matroid* properties
- Quality of the solution obtained with the greedy algorithm will meet goal and utility function policy requirements

18

Constraint based framework

Objective function / Constraints	Linear	Submodular	Unrestricted
Matroid	Optimal Utility Function	½ approximation Goal	No guarantees Action
K-extendible	1/k approximation Goal	1/k+1 approximation Goal	No guarantees Action
Unrestricted	No guarantees Action	No guarantees Action	No guarantees Action

19

Resource Allocation in Distributed Systems Objective Function Based

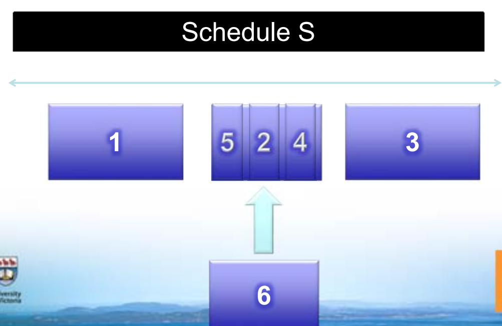
- We are given
  - A set V = { 1, 2, 3.. M} of M servers
  - A set R = {1,2,3,… l} resources
  - Further more we assume that every resource type such as memory , CPU or bandwidth are split into many blocks of fixed size so that one or more such blocks can be assigned to each server.
- Goal: Maximize the sum of the throughputs of the servers
- Constraints
  - Every resource is allocated to at most one server

32

Linear objective function

22

Processing time — No condition



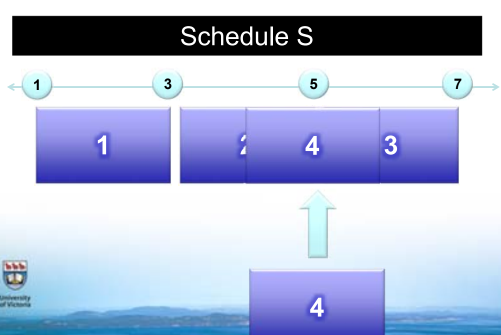
23

General — Action policy

- When processing times are arbitrary:
  - Constraint set does not have nice structure
  - No theoretical guarantees for the performance of the greedy algorithm
  - It satisfies the expectations of an action policy

24

Processing time — All equal



25

2-extendible property—Goal policy

- Processing times are equal

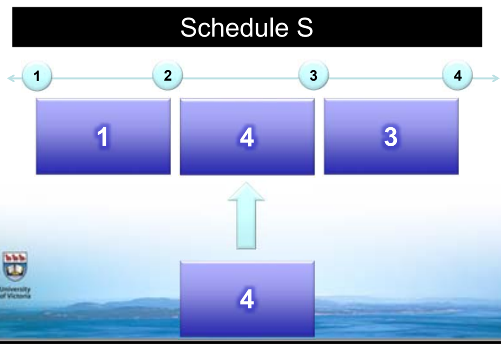
➡

- Constraint set satisfies the 2-extendible property
- Applying Mestre's result the greedy technique gives ½ approximation
- Approximation algorithms are the mathematical equivalent of goal policies

J. Mestre: Greedy in approximation algorithms. In: Proc. 14th Annual European Symposium on Algorithms (ESA), pp. 528-539 (2006)

26

Processing time — Unit time



1-extendible or matroid property Utility function policy

- Processing times are unit times
- Constraint set forms a matroid
- According to Edmonds the Greedy algorithm produces an optimal solution
- Satisfies the requirements of a utility function policy

J. Edmonds: Matroids and the Greedy algorithm. Mathematical Programming Studies, 1(1):27-36 (1971)

Scheduling on Distributed Set of Clouds

Formal Problem Description

- P jobs $J_1, \ldots, J_p$ needs to be scheduled on the m clouds $C_1, \ldots, C_m$. Each cloud has the following
  - Deployment Configurations (DC): n_i
  - Each DC : $\{J_1, \ldots, J_p\}$
  - Revenue: r_{ij}
- Goal : Is to choose a Deployment Strategy (DS) that maximizes the total revenue.The total revenue of all the clouds schedule is the sum of the revenues of all the DC in the schedule.
- Constraints
  - Choose at most one DC from each cloud
  - Each DS selected has each job appearing at most once across all clouds

Observations

- Objective Function is Linear
- In General – exchange not satisfied
- If deployment configurations are of size at most s, we get (s+1)-extendible system
- If we remove a constraint in the problem, the constraint set forms a matroid

Objective function based framework

- Assume that the constraint set of the underlying optimization problem satisfies the Matroid property
- Then vary the objective function
- Add structure to the objective function to make it submodular and even linear
- Quality of the solution obtained with the greedy algorithm meets goal and utility function policy requirements

Objective function based framework

Constraints \ Objective function	Linear	Submodular	Unrestricted
Matroid	Optimal Utility Function	½ approximation Goal	No guarantees Action
K-extendible	1/k approximation Goal	1/k+1 approximation Goal	No guarantees Action
Unrestricted	No guarantees Action	No guarantees Action	No guarantees Action

Contributions

1
- Mathematical formulation for the three policy types
- First precise characterization of goal policies for optimization problems

2
- Mathematical framework to add structure to optimization problems to progressively increase the solution quality when using the greedy algorithm

3
- Framework to optimization problems in the realm of self-adaptive and self-managing systems

S. Balasubramanian *et. al.*: Characterizing Problems for Realizing Policies in Self-Adaptive and Self-Managing Systems, SEAMS 2011