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

Sowmya Balasubramanian
Ron Desmarais
Hausi A. Müller
Ulrike Stege
S. Venkatesh

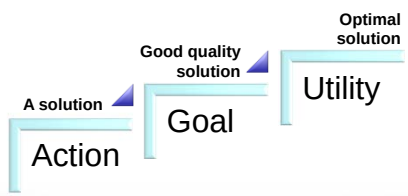
University of Victoria

SEAMS 2011
Honolulu, Hawaii, USA
May 2011

Outline

- Goals and related work
- Characterizing policy-based optimization problems using the *Greedy algorithm*
- Mathematical framework to add structure to problems to guarantee solution quality
- SEAMS studies

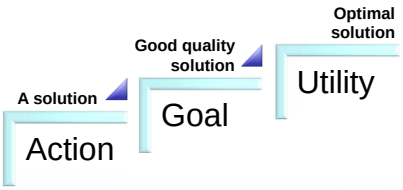
Policy framework by Kephart & Walsh



Algorithms increase in sophistication

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

Our approach



Add problem structure for Greedy algorithm

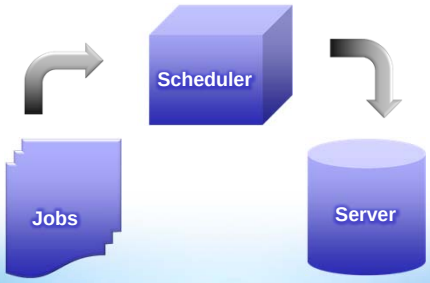
Our research question

- Is it possible to add structure to a SEAMS optimization problem so that the resulting solution can meet requirements of goal and utility function policies?

Our main contribution

- Is it possible to add structure to a SEAMS optimization problem so that the resulting solution 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 obtained using the Greedy algorithm

A typical SEAMS problem
Data center scheduling



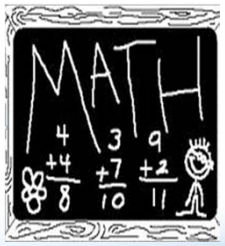
Data center scheduling problem

- 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.

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	No guarantees Action	No guarantees Action
Unrestricted	No guarantees Action	No guarantees Action	No guarantees Action

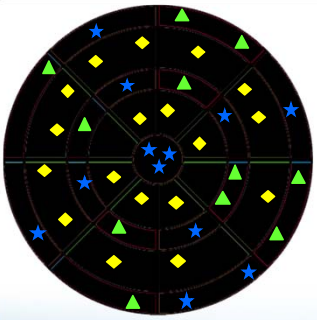
How to use our handbook

- Handbook for SEAMS optimization problems
- 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

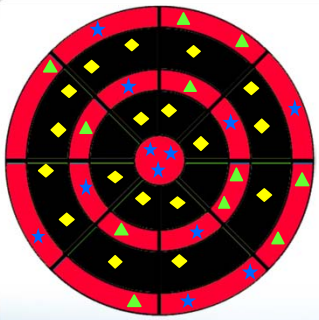
- Regions represent solution qualities
- Aim for high quality regions

Action dart board



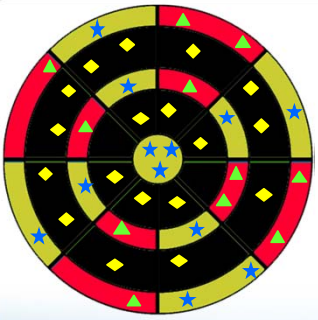
Legend	
★	Optimal solution
▲	Good solution
◆	A solution

Goal dart board



Legend	
★	Optimal solution
▲	Good solution
◆	A solution

Utility function dart board



Legend	
★	Optimal solution
▲	Good solution
◆	A solution

SEAMS optimization case studies

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

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

Constraint based framework

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

Data center scheduling problem

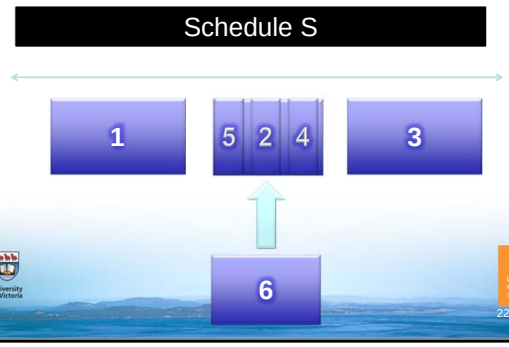
- Given a set of n Jobs $J_1, \ldots, J_n$ each with the following parameters:
  - Arrival time: A_i
  - Deadline: D_i
  - Processing time: P_i
  - Revenue: R_i
- Schedule the jobs on a single server so that the total revenue is maximized.
- The total revenue of a schedule is the sum of the revenues of the jobs processed in the schedule.

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

Linear objective function

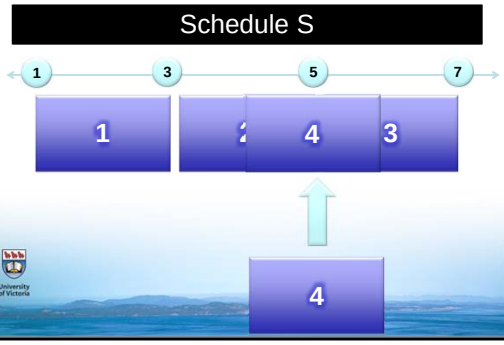
Processing time — No condition



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

Processing time — All equal

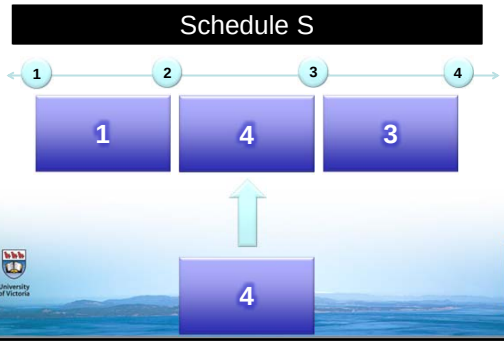


2-extendible property—Goal policy

- 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)

Processing time — Unit time



1-extendible or matroid property Utility function policy

- Processing times are unit times
- Constraint set forms a Matroid
- According to Edmond 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)

Objective function 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 from to make it submodular and even linear
- Quality of the solution obtained with the greedy algorithm meets goal and utility function policy requirements

Objectives based framework

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

Contributions

1. Mathematical formulation for the three policy types
   - In particular, first precise characterization of goal policies for optimization problem
2. Introducing a mathematical framework to add structure to optimization problems to progressively increase the solution quality when using the greedy algorithm
3. Applying our framework to optimization problems in the realm of self-adaptive and self-managing systems

Ongoing work

- Experimental evaluation of the Greedy technique to study its average case behaviour
- To extend this work to similar mathematical structures for other algorithmic techniques including *Dynamic Programming*
- Categorize SEAMS optimization problems according to our handbook

University of Victoria

University of Victoria Engineering

31-33

Summary

University of Victoria

University of Victoria Engineering

32-33

University of Victoria

33-33