





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



Venkatesh Srinivasan  University of Victoria
July 5, 2013

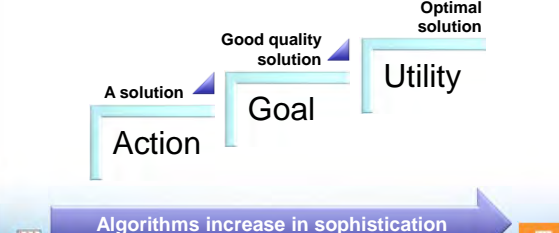
Outline

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



 

1

Policy framework by Kephart & Walsh

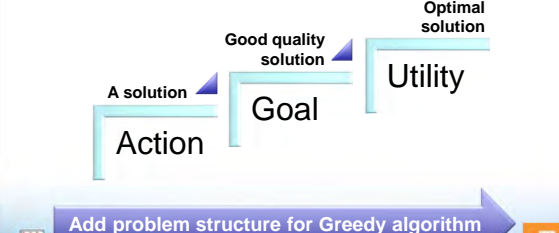




J. Kephart, W. Walsh: An AI perspective on autonomic computing policies. In: Proc. 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 3-12 (2004)

2



Our approach

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?

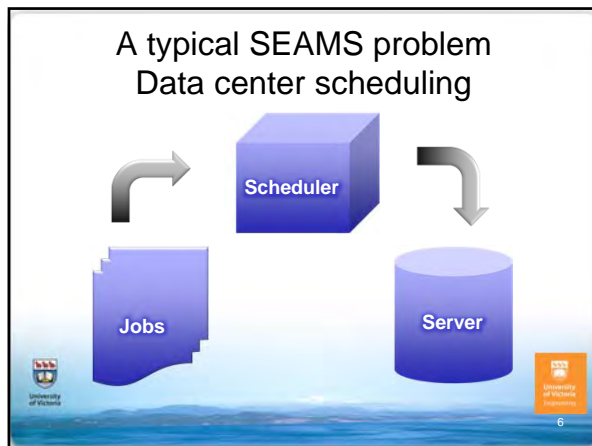
4

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



Data center scheduling problem

- Given a set of n Jobs J_1, \dots, 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.

University of Victoria

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

University of Victoria

8

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

University of Victoria

9

Handbook for designing policy-driven optimization strategies

Objective function \ Constraints	Linear	Submodular	Unrestricted
Matroid	Optimal	$\frac{1}{2}$ approximation	No guarantees
K-extendible	Utility Function	Goal	Action
	$\frac{1}{k}$ approximation	$\frac{1}{k+1}$ approximation	No guarantees
	Goal	Goal	Action
Unrestricted	No guarantees	No guarantees	No guarantees
	Action	Action	Action

University of Victoria

10

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

University of Victoria

11

Metaphor Solution quality dartboard


- Regions represent solution qualities
- Aim for high quality regions



University of Victoria logo and SEAMS logo are present in the bottom left and right corners respectively.

12

Action dart board



Legend	
★	Optimal solution
▲	Good solution
◆	A solution

University of Victoria logo and SEAMS logo are present in the bottom left and right corners respectively.

13

Goal dart board

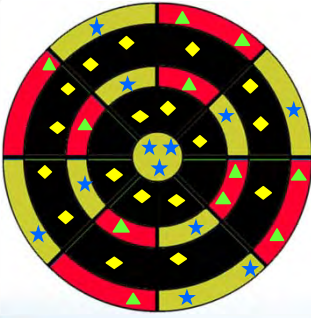


Legend	
★	Optimal solution
▲	Good solution
◆	A solution

University of Victoria logo and SEAMS logo are present in the bottom left and right corners respectively.

14

Utility function dart board



Legend	
★	Optimal solution
▲	Good solution
◆	A solution

University of Victoria logo and SEAMS logo are present in the bottom left and right corners respectively.

15

SEAMS applications

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

University of Victoria logo and SEAMS logo are present in the bottom left and right corners respectively.

16

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

University of Victoria logo and SEAMS logo are present in the bottom left and right corners respectively.

17

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	1/2 approximation	No guarantees
K-extendible	1/k approximation	1/k+1 approximation	No guarantees
Unrestricted	No guarantees	No guarantees	No guarantees
	Utility Function	Goal	Action
	Goal	Action	Action
	Action	Action	Action



19

Data center scheduling problem

- Given a set of n Jobs J_1, \dots, 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.



20

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

Linear objective function



22

Processing time — No condition



23

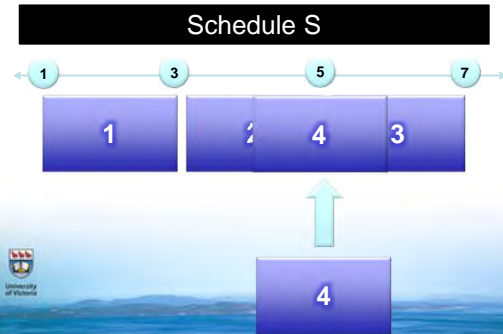
General — Action policy

- 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 $\frac{1}{2}$ approximation
- Approximation algorithms are the mathematical equivalent of goal policies

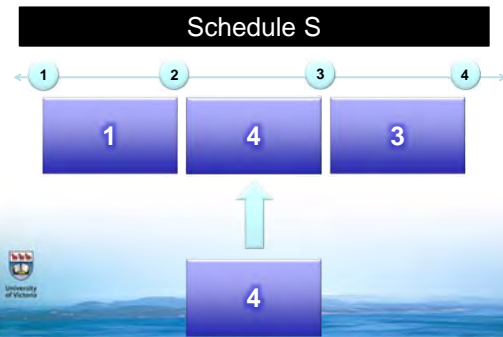


J. Mestre: Greedy in approximation algorithms. In: Proc. 14th Annual European Symposium on Algorithms (ESA), pp. 528-539 (2006)



26

Processing time — Unit time



27

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)



28

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



29

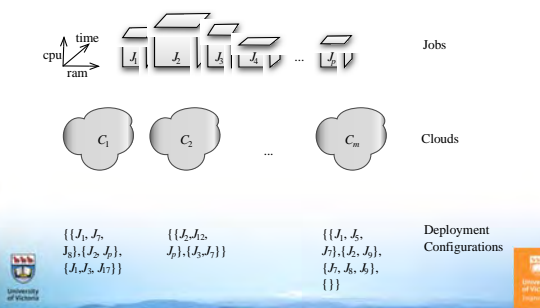
A Case Study



Cloud Scheduling – Part 4

30

Scheduling on distributed set of clouds





31

Formal Problem Description



- P jobs J_1, \dots, J_p needs to be scheduled on the m clouds C_1, \dots, C_m . Each cloud has the following
 - Deployment Configurations (DC): n_i
 - Each DC : $\{J_1, \dots, J_p\}$
 - 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 atmost one DC from each cloud
 - Each DS selected has each job appearing at most once across all clouds

32

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

33

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

34

Objective function based framework

Objective function \ Constraints	Linear	Submodular	Unrestricted
Matroid	Optimal	1/2 approximation	No guarantees
K-extendible	Utility Function	Goal	Action
Unrestricted	1/k approximation	1/k+1 approximation	No guarantees
	Goal	Goal	Action
	No guarantees	No guarantees	No guarantees
	Action	Action	Action

35

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

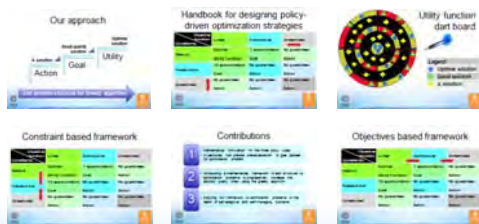
36

Future 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

37

Summary



38

Thank You!

Questions?

University of Victoria

39

BACK UP SLIDES



Resource Allocation in Distributed Systems Objective Function Based

- We are given
 - A set $V = \{1, 2, 3, \dots, M\}$ of M servers
 - A set $R = \{1, 2, 3, \dots, I\}$ 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 : Is to maximize the sum of the throughputs of the servers.
- Constraints
 - Every resource is allocated to at most one server



32

Verification

- $U = L \text{ Resources} \times M \text{ servers}$
- $U = 2 \text{ Resources} \times 3 \text{ Servers}$
- $U = \{1,2\} \times \{1, 2, 3\}$
- $\{ (1,1), (1,2), (1,3), (2,1), (2,2), (2,3) \}$
- $2^U =$ set of all possible subsets of U . If n is the number of tuples in U then this will have 2^n elements.
- $\{ \{ (1,1) \}, \{ (1,2) \}, \{ (1,1), (1,2) \}, \dots \}$
- $F =$ set of all possible allocations satisfying the constraints of the given problem. Resource l is assigned to 0 or 1 server.



32

Verification (cont)

- $F = \{ \Phi, \{ (1,2), (2,3) \}, \{ (1,2), (2,2) \}, \{ (1,3) \}, \dots \}$
 - $A = \{ (1,2), (2,3) \}$
 - $B = \{ (1,3), (2,2) \}$
 - $C = \{ (1,3) \}$
- Downward Closure : If A satisfies the constraints then any sub allocation of $A = (2,3)$ will also satisfy the constraint and will be in F
- Augmentation : If we take A and C where $|A| > |C|$. Then we can add $(2,3)$ to C and it will be a valid allocation.



32

Linear Objective Function

- The objective function is linear since the total revenue R of a schedule S is defined as sum of the individual revenues of the jobs processed in the schedule.



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.



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.



Role of Policy

- Autonomic Computing is a vision of "Self-Managing" systems.
- Systems managing their own behavior with high level objectives (Policies) specified by the human administrators.
- Autonomously perform many tasks related to configuration, optimization, healing, and protection.

"Policy is any type of formal behavioral guide"

"Policies are a form of guidance used to determine decisions and actions"



Greedy Algorithm

- Sort the jobs based on the Revenue R_i .
- Start with the empty schedule. Add a new job to the current job, if feasible. The choice of start time can be arbitrary.



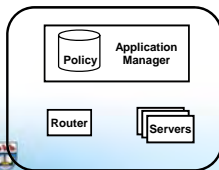
Evaluation

- Case study to extensively analyze optimization problems in self-managing systems such as
 - Resource allocation in distributed systems
 - Resource allocation in QoS service management
 - Data center based scheduling problem
 - SLA profit optimization

Experimental evaluation of the greedy technique to study its average case



Data Centre Example



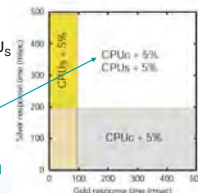
Application Environment i

- To provide good service to the 2 transaction classes namely
 - GOLD
 - SILVER
- Policy are discussed at 3 levels
 - **AM to apportion existing resources among the gold and silver transaction classes**
 - AM's resource requests to the Resource Arbiter
 - Resource Arbiter allocation of Resources to the AM



Decision Making using Action Policy : Application Manager Allocating Resources

- G. IF ($RT_G > 100$ msec) THEN (Increase CPU_G by 5%)
- S. IF ($RT_S > 200$ msec) THEN (Increase CPU_S by 5%)



- Conflict will arise when the CPU is fully utilized
 - CPU utilization as a part of the condition
 - Assigning Priorities
- This makes policy sets very complex.

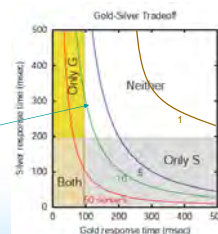


Decision Making using Goal Policy : Application Manager Allocating Resources

- Takes the form of performance targets.
- Represents the desired response time, as opposed to the current response times.

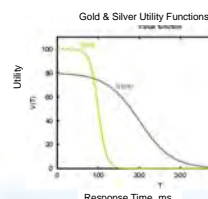
- G: $RT_G < 100$ msec
- S: $RT_S < 200$ msec

- Conflicts arise when the desired state is not a feasible state.
- A naive approach is to give up lower priority goals.
- However the conflict resolution policy is not quite clear-cut as one might expect.

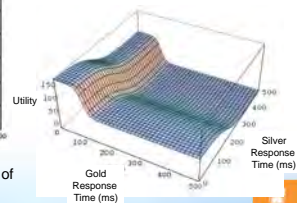


Policies contain no hint about how the goals are to be attained. The system must employ other mechanisms to translate goals into actions.

Decision Making using Utility function Policy : Application Manager Allocating Resources



$$U(RT_G, RT_S) = U_G(RT_G) + U_S(RT_S)$$



- Utility policies are softened version of individual goals
- Two utility functions can be combined a single utility number of ways e.g., simple summing of individual utility functions



Mathematical structure?

- Any optimization problem has two components
 - Objective Functions
 - Set of Constraints
- Structure means to fix one part of the problem and vary the other.
- Using this we have created two frameworks.



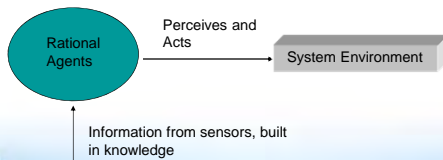
Constraint Based Framework (Vary Constraint Set)

- Objective function is linear.
- Gradually structure the constraint set from general to k-extendibility and then to matroid.
- The quality of the solution obtained by the greedy technique meet the specification of action-goal-utility function policy.



Artificial Intelligence

- AI is fundamentally concerned with design of rational agents (machine, or software that makes its own decisions)

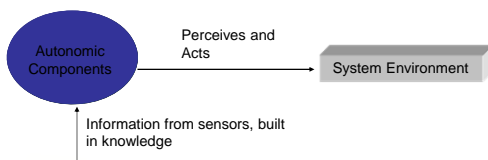


Three Approaches to Rational Agents Design

- Reflex agents – Specify “what to do” under the current condition
 - Uses *if - then action* rules.
 - Rationality : compiled by the designer or pre-computed.
- Goal based agents
 - Moderate Rationality : effectively decides which action to take to achieve a particular goal.
 - Provides more flexibility than Reflex agents
- Utility based agents
 - High rationality : chooses actions to maximize its utility functions.
 - Finer distinction between desirability of different states than goal based.



An AI Perspective on Autonomic Computing Policies Kephart & Walsh – POLICY 2004



Autonomic Components can be designed as rational agents which have built in policies to guide them.

