Improving Energy-Bound Application Effectiveness through Redundancy and Approximation

Liu Liu[1], Timothy Yong[1], Jonathan Risinger[1], Sibren Isaacman[2], Abhishek Bhattacharjee[1], and Ulrich Kremer[1]
Department of Computer Science
[1] Rutgers University / [2] Loyola University

EEL Lab
Computer Science

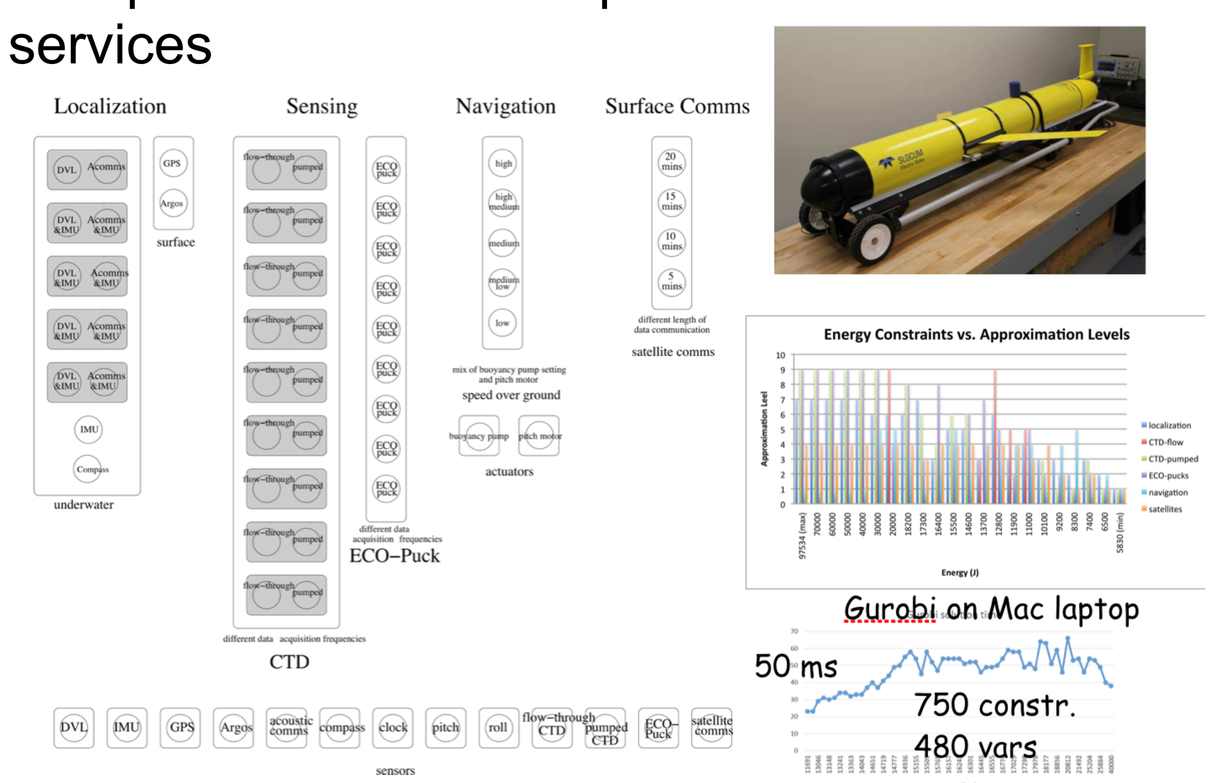
Target Applications

1. Composed of sets of collaborating services (producer-consumer, resource sharing)
2. Each individual service may provide discrete levels of quality vs. energy (resource) tradeoffs.
3. Nothing is free! Want best overall application outcome within a given energy budget.

AUVs — Service robots — Smart phones — Networks in underdeveloped regions

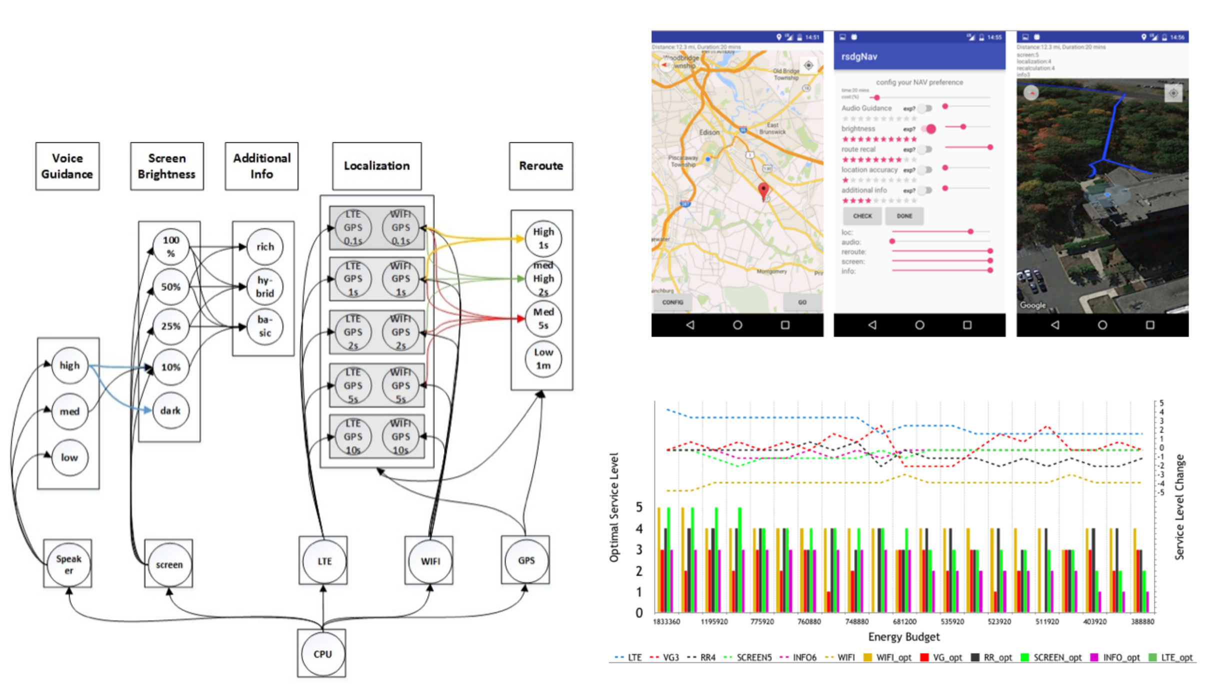
RSDG Glider

Multiple sensors and dependencies between services

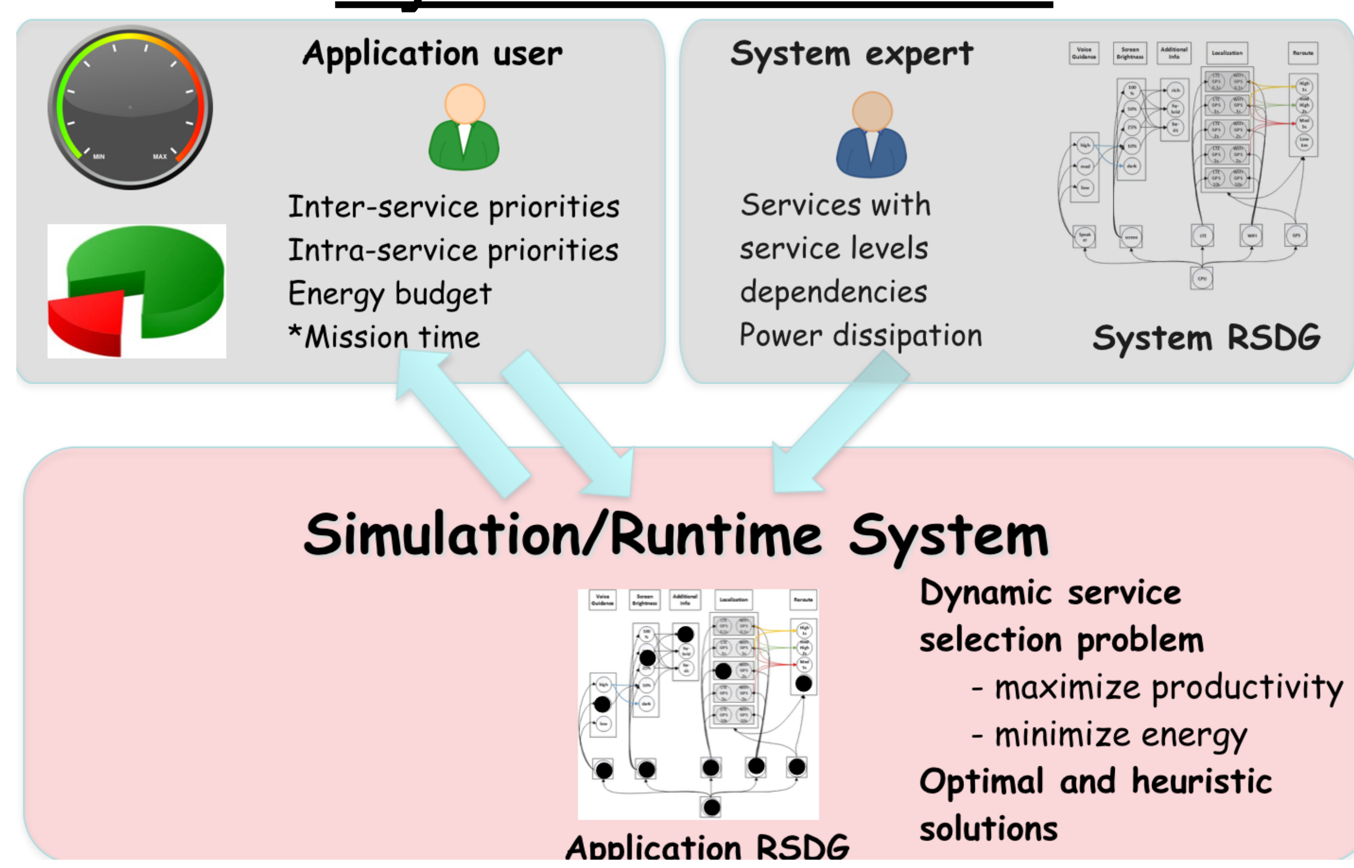


RSDG NavApp

Navigation application on Android that balances multiple services to meet the energy requirement.



System Overview

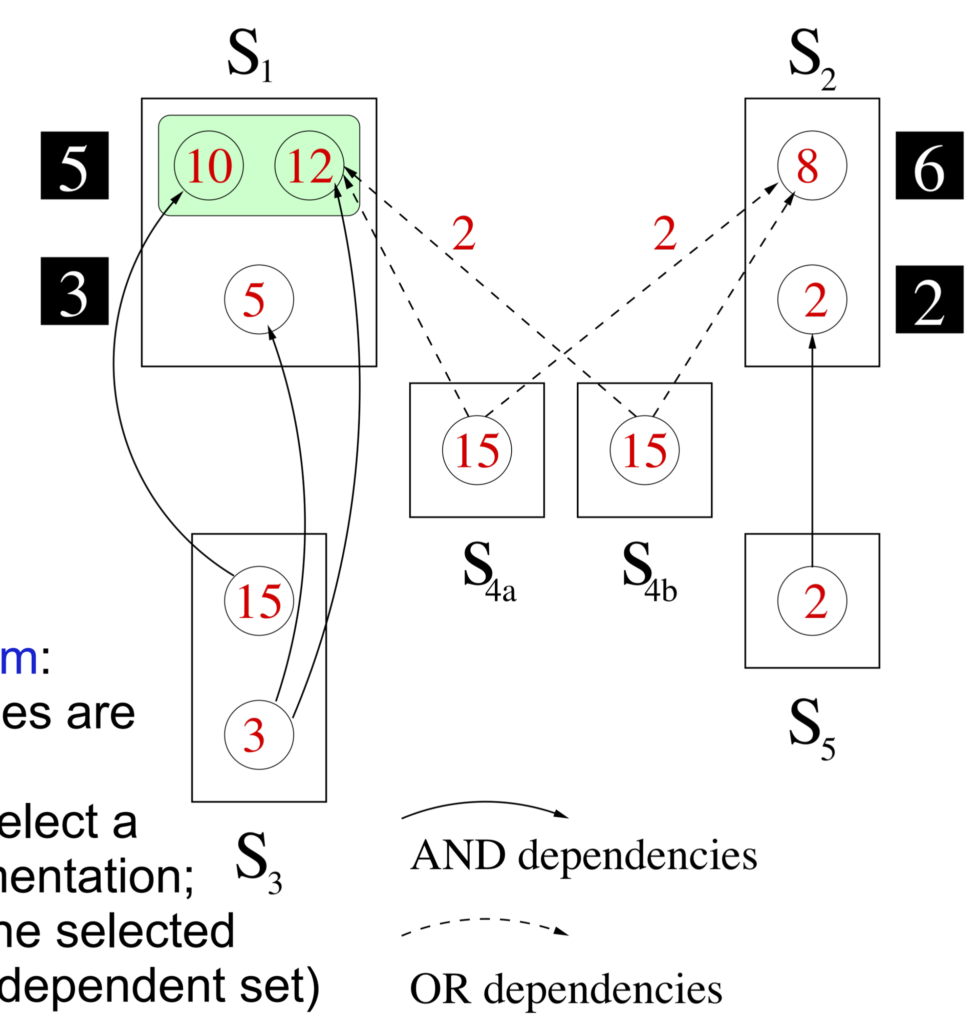


Design Foundation

3 types of redundancy:
- approximation
- implementation
- replication

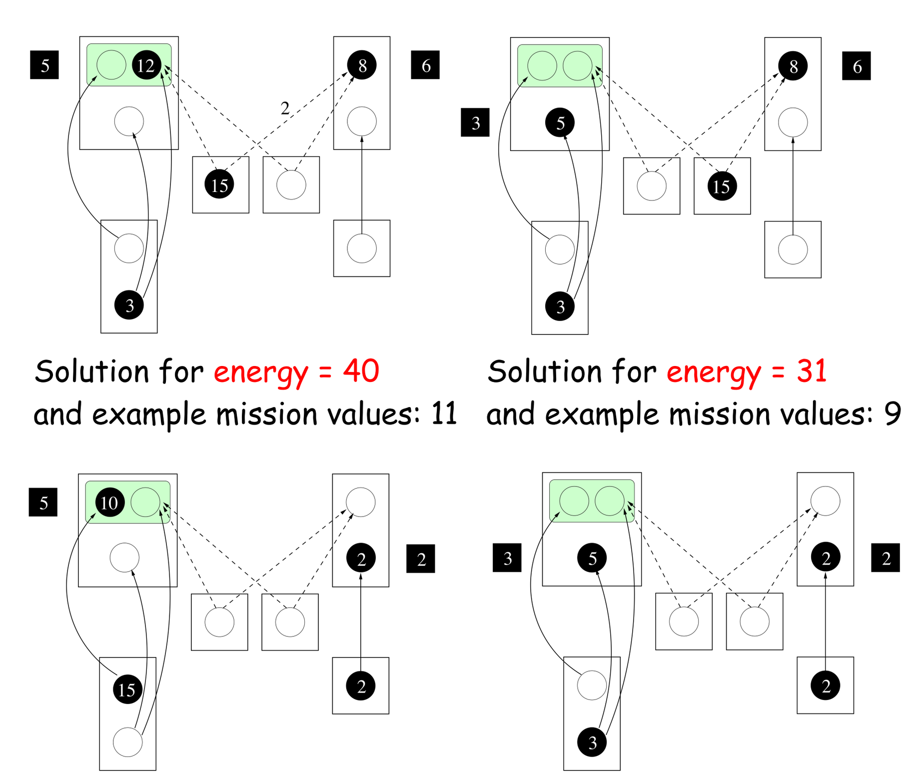
mission value

energy consumption



RSDG service selection problem:
- Services with mission values are considered (user) critical
- For each critical service, select a single service level and implementation; do the same for each service the selected service depends on (transitive dependent set)
- Sum of mission values of critical services is maximal under given energy constrains; energy is minimized for this maximal mission value

Productivity Profile

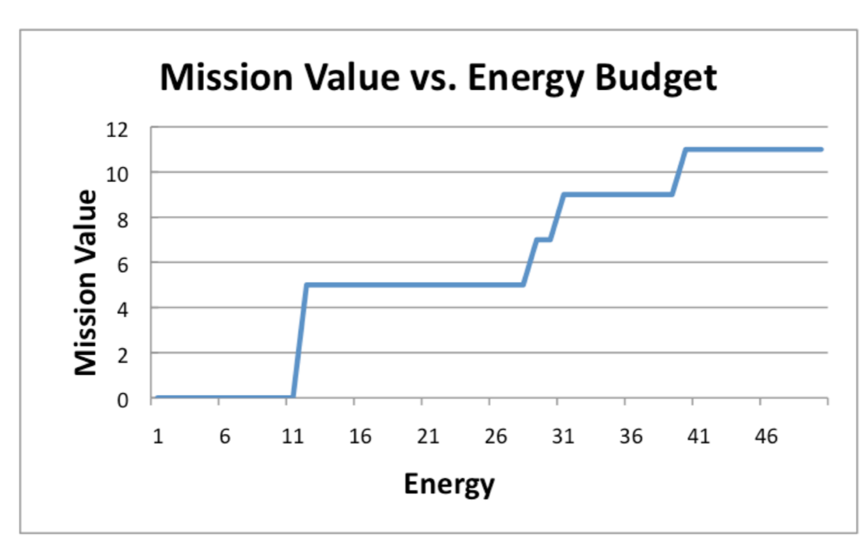


Solution for energy = 40 and example mission values: 11

Solution for energy = 31 and example mission values: 9

Solution for energy = 29 and example mission values: 7

Solution for energy = 12 and example mission values: 5



Optimal solution balances service qualities across the entire application; Solution seems sometimes non-intuitive.
Solution is NP-complete; Proof: Reduction from 3SAT. Use 0-1 integer programming formulation and gurobi, cplex, or lp_solve
Simulations can provide feedback during tradeoff space exploration via the Productivity Profile:

Application Analysis

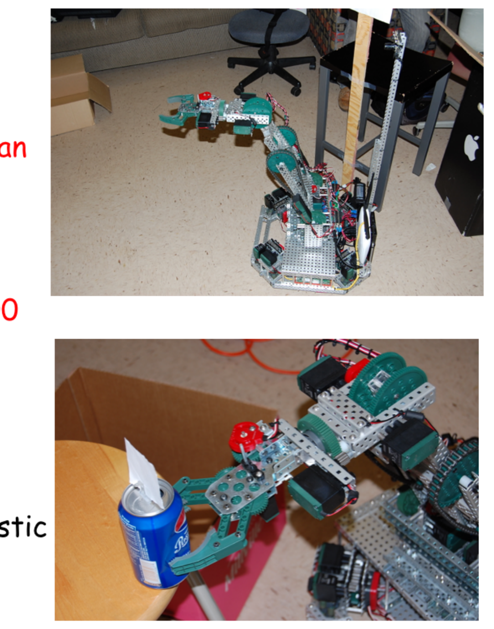
Energy Cost vs. Overhead vs. Benefit

Simulation

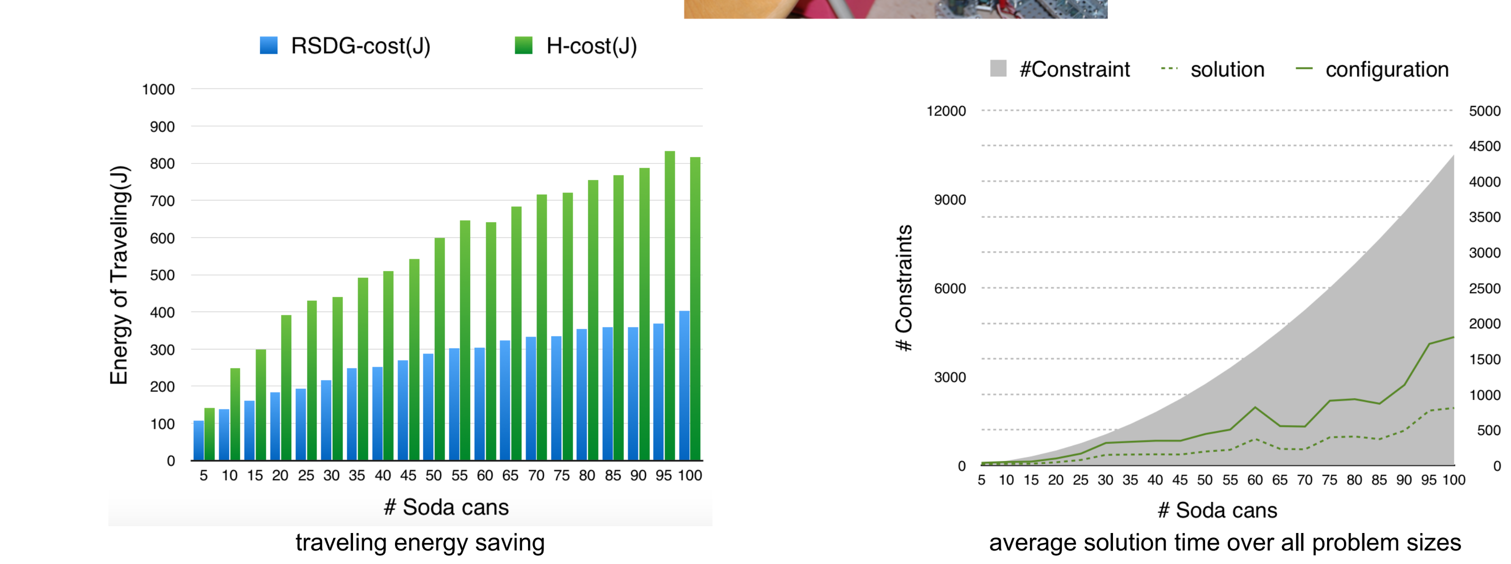
Energy Cost:
- traveling / driving : 5 Joules for 10 cm
- grabbing / lifting : 10 Joules per soda can

Benefit:
- number of soda cans : between 5 and 100
- mission values (preferred type of soft drinks) : Coke 5 vs. Pepsi 1

Overhead (path planning):
- Our RSDG approach vs. a "clever" heuristic

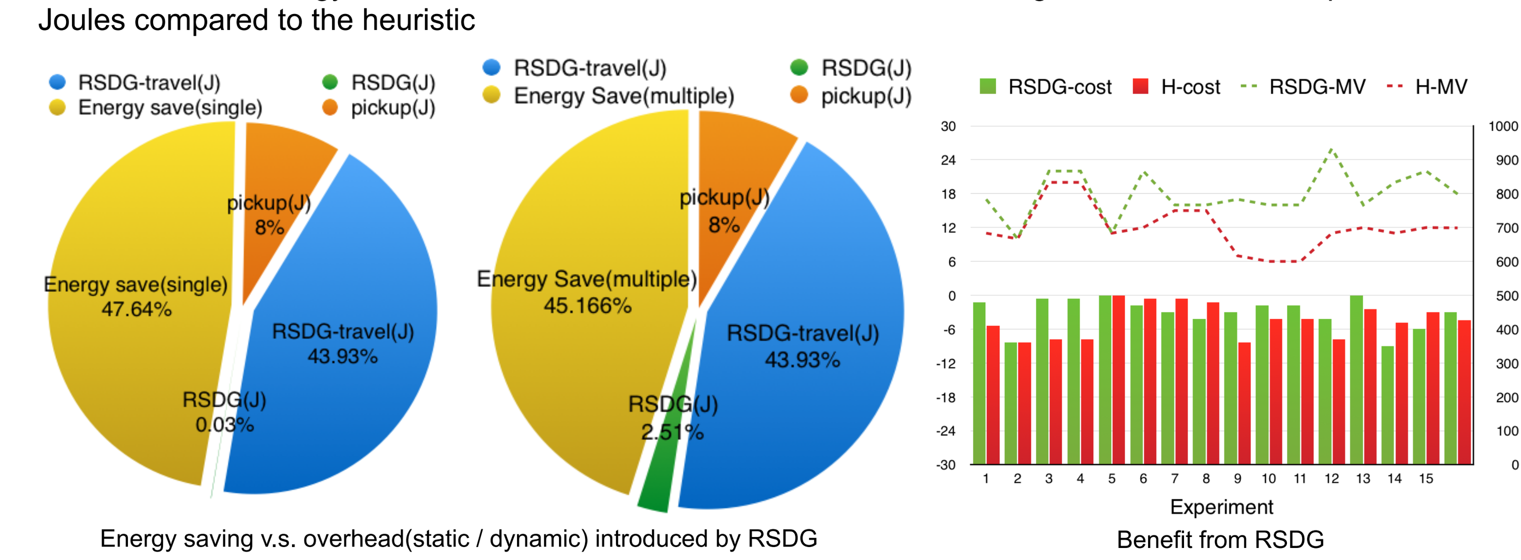


10 x 10 meters
5 - 100 soda cans
Randomly generated locations
traveling cost: 50J / m



Given sufficient energy, RSDG's solution saved ~50.4% Joules compared to the heuristic

A full configuration consists of 2 problem solving.



Static : Configure the system once for all at the beginning.
Dynamic: Re-configure the system 1 time per pickup.
Saved more than 45% energy by introducing less than 2.5%.

Given 10 soda cans with Budget = 500J in 15 experiments.8 CansValue (~64%) more than heuristic.

Conclusion

Conclusion:
A new framework that allows the programmer to easily describe an adaptive system that balances the services configurations across the entire system to produce highest productivity with lowest cost within user-specific budget.

Related Work:
Dynamic Knobs for Responsive Power-Aware Computing [ASPLOS-11]:
Improving performance and power savings by dynamically adjusting the parameters to "trade off the accuracy of computation in return for reductions in resources."
*problem: accuracy could not be a measurement for quality in most of the time. (Users' preference(priority) got ignored)
JouleGuard: Energy Guarantees for Approximate Applications[SOSP-15]:
Maximizing accuracy within energy budget by "dividing the problem into two sub-problems, configuring hardware system to be energy efficient and dynamically adjusting program to manage performance."
*problem: ignored the dependencies.(smaller search space)
CAreDroid: Adaptation Framework for Android Context-Aware Applications[MobiCom-15]:
Improving efficiency of context-aware applications by "monitoring the context of the physical environment and intercepts calls to sensitive methods".
*problem: not being able to handle dependencies other than physical context.(single layer RSDG)

Future Work:
-more application examples
-better tools that support easier construction for RSDG
-better tools for users to explore the tradeoff spaces
-better execution model
-optimization on 0-1 problem's constraints
-reducing the RSDG overhead