MADRAS: Multiagent Distributed Resource Allocation Simulator

Gijs Kruitbosch, Nadya Peek
based on work with Hylke Buisman and Ulle Endriss
University of Amsterdam

3rd MARA Get-Together, 5-6 June 2008

Introduction

Resource allocation is an important issue in multiagent systems.

Finding the optimal allocation is difficult.

Distributed approaches share the computational load.

How do we predict outcomes of distributed negotiation?

To gain more insight, we try to simulate such distributed resource allocation.

Overview

- Framework, notation
- System overview and usage:
  - Generating scenarios
  - Running experiments
  - Visualizations
- Example experiments and results
- Conclusion

Framework

All agents: $\mathcal{A} = \{a_1, a_2, ..., a_n\}$, all resources: $\mathcal{R} = \{r_1, r_2, ..., r_m\}$.

An allocation A is a division of the resources amongst the agents.

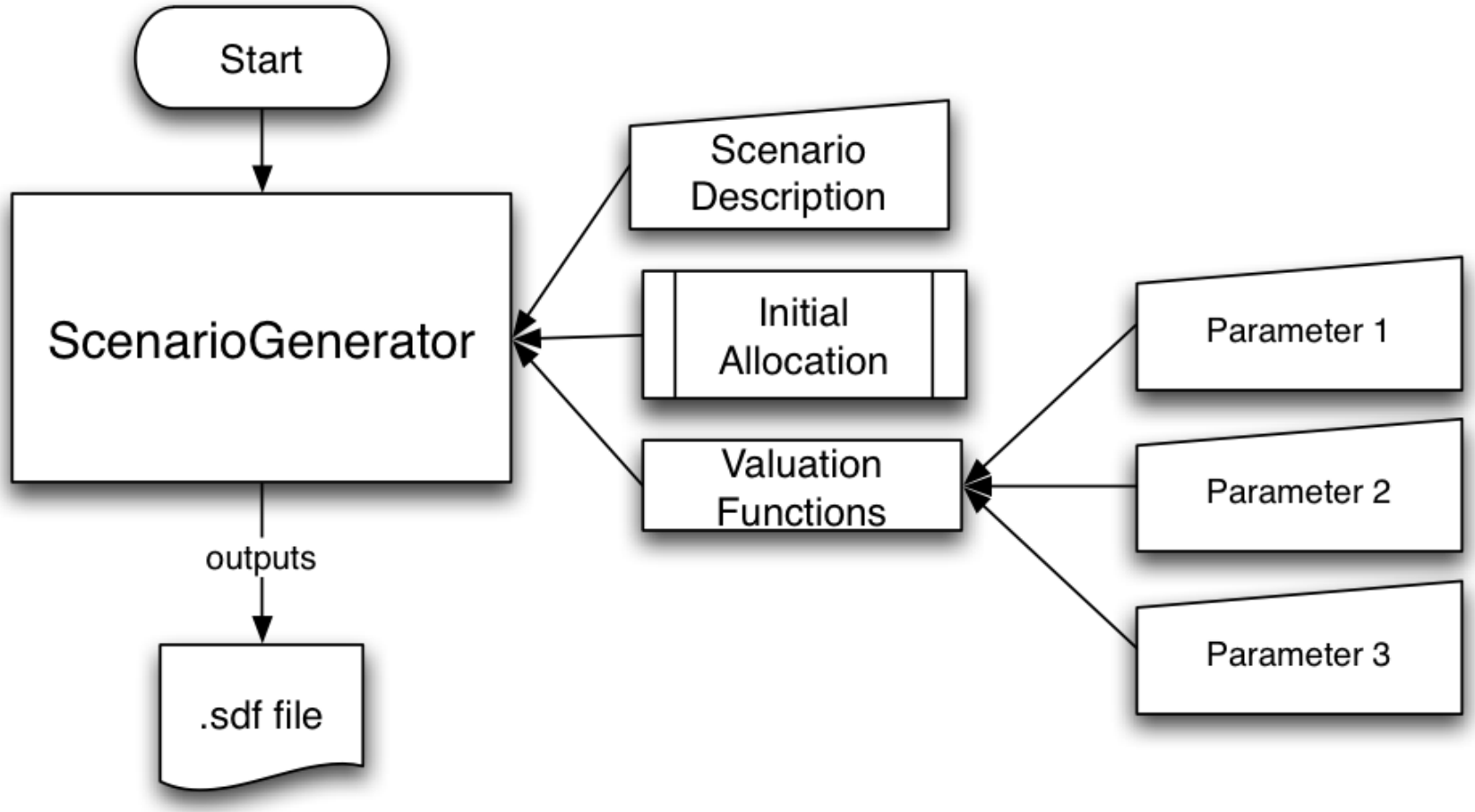
A deal δ consists of two allocations (A, A').

Each agent i has a valuation function $v_i : 2^{\mathcal{R}} \rightarrow \mathbb{R}$ to model their preferences. This is done with a logic based language, where each resource $r \in \mathcal{R}$ is used as a propositional variable. Agents express their preferences by giving weights to propositional formulas, e.g.

- $\{(hammer, 10), (nail, 1), (hammer \wedge nail, 3)\}$
- $\{(theatre, 10), (football, 12), (theatre \wedge football, -9)\}$

While simulating resource allocation we are interested in

- different forms of social welfare: egalitaritian, elitist, utilitarian
- envy within the society of agents


Start
ScenarioGenerator
outputs
.sdf file
Scenario Description
Initial Allocation
Valuation Functions
Parameter 1
Parameter 2
Parameter 3

Generating Valuation Functions

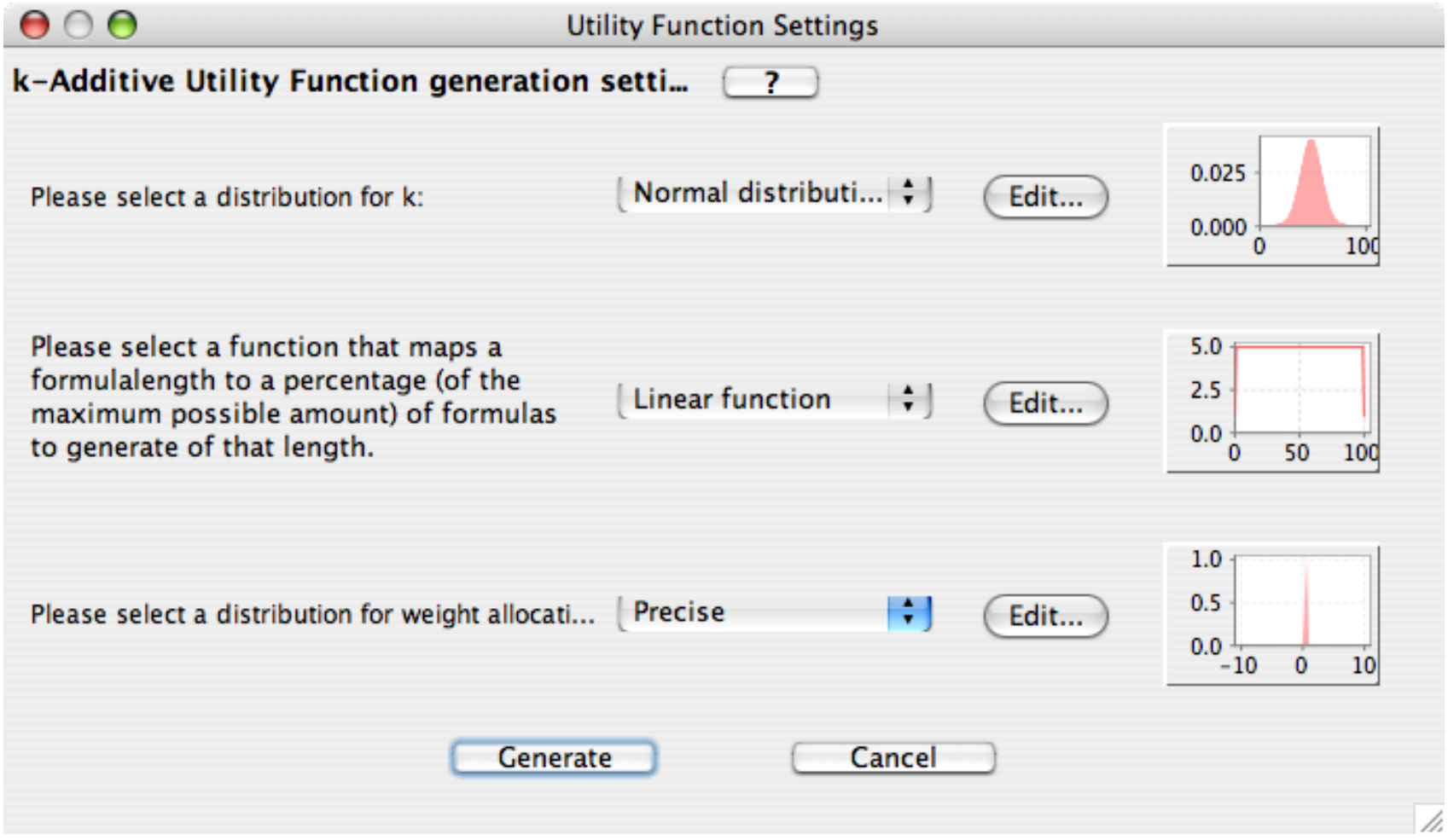
For the automatic generation of valuation functions, we have restricted ourselves to k-additive valuations, which are defined by weighted conjunctions of positive literals of length $\leq k$.

Example: $(r_1 \wedge r_2 \wedge r_5 \wedge r_9, 12)$ is within $k = 4$.

Parameters for the generation of k-additive valuation functions:

- Distribution for k: where k is the maximum size of the set of resources referenced by one goal.
- Goal length to count mapping: where we determine how many formulas will be generated of a particular length.
- Weight variation: how much should an agent like each goal defined in its valuation function?

MADRAS: Generating Valuation Functions



Making Deals in MADRAS

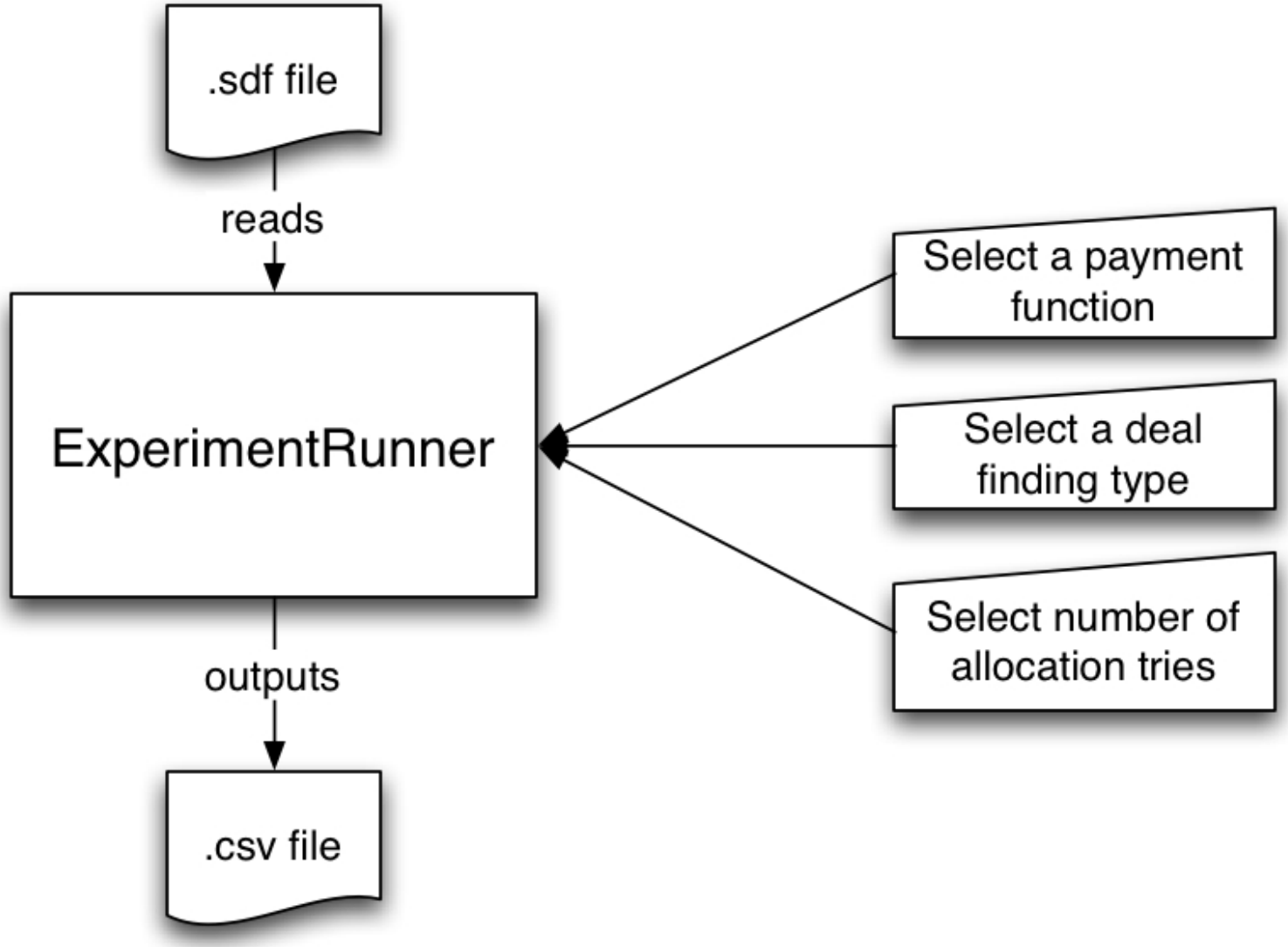
MADRAS can compare the effects of different negotiation policies on the society's social welfare. In particular, we compared two deal making mechanisms.

1-resource deals

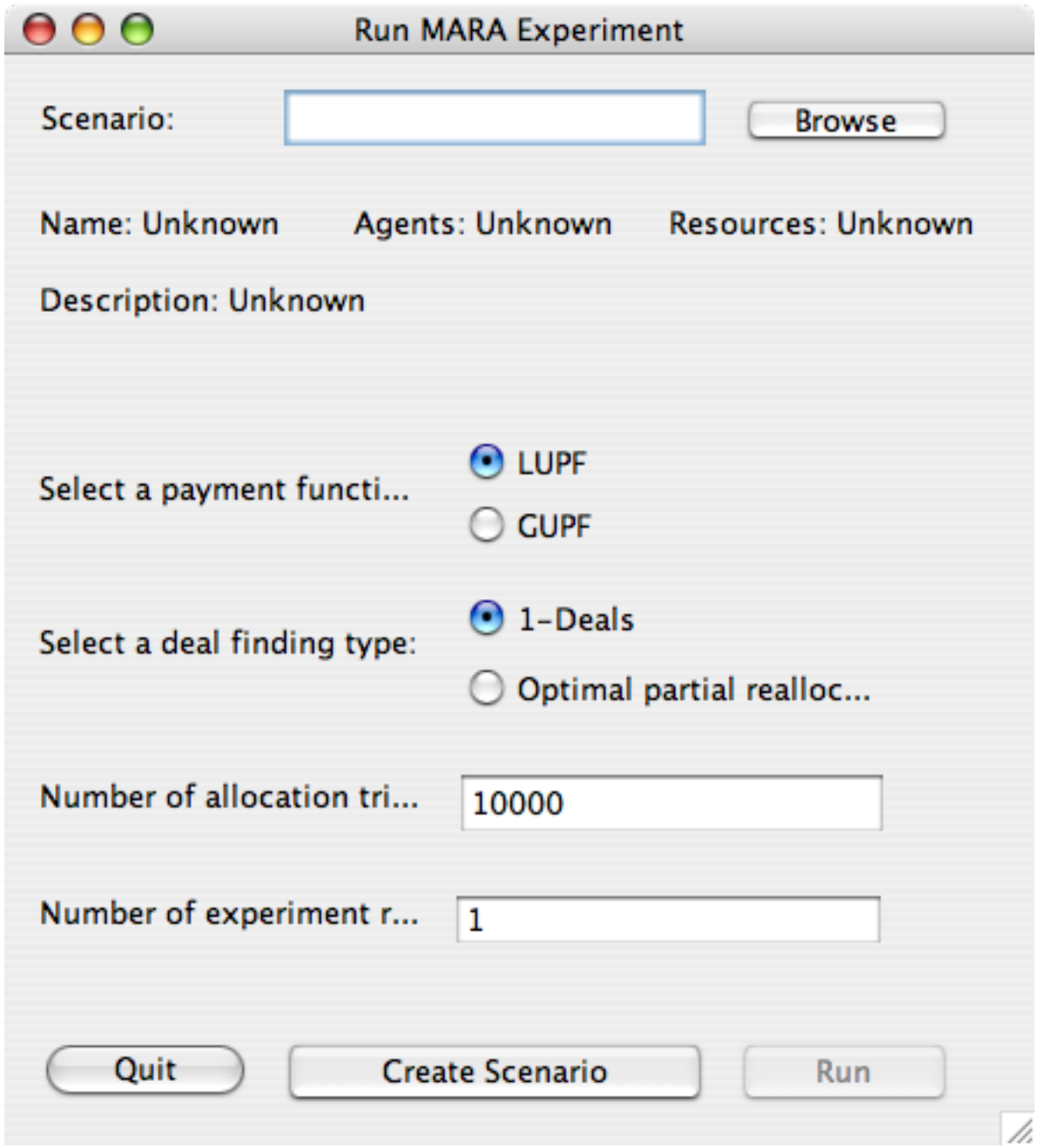
- Randomly select a pair of agents
- Check whether there is a resource for which it is individually rational for the agents to trade.

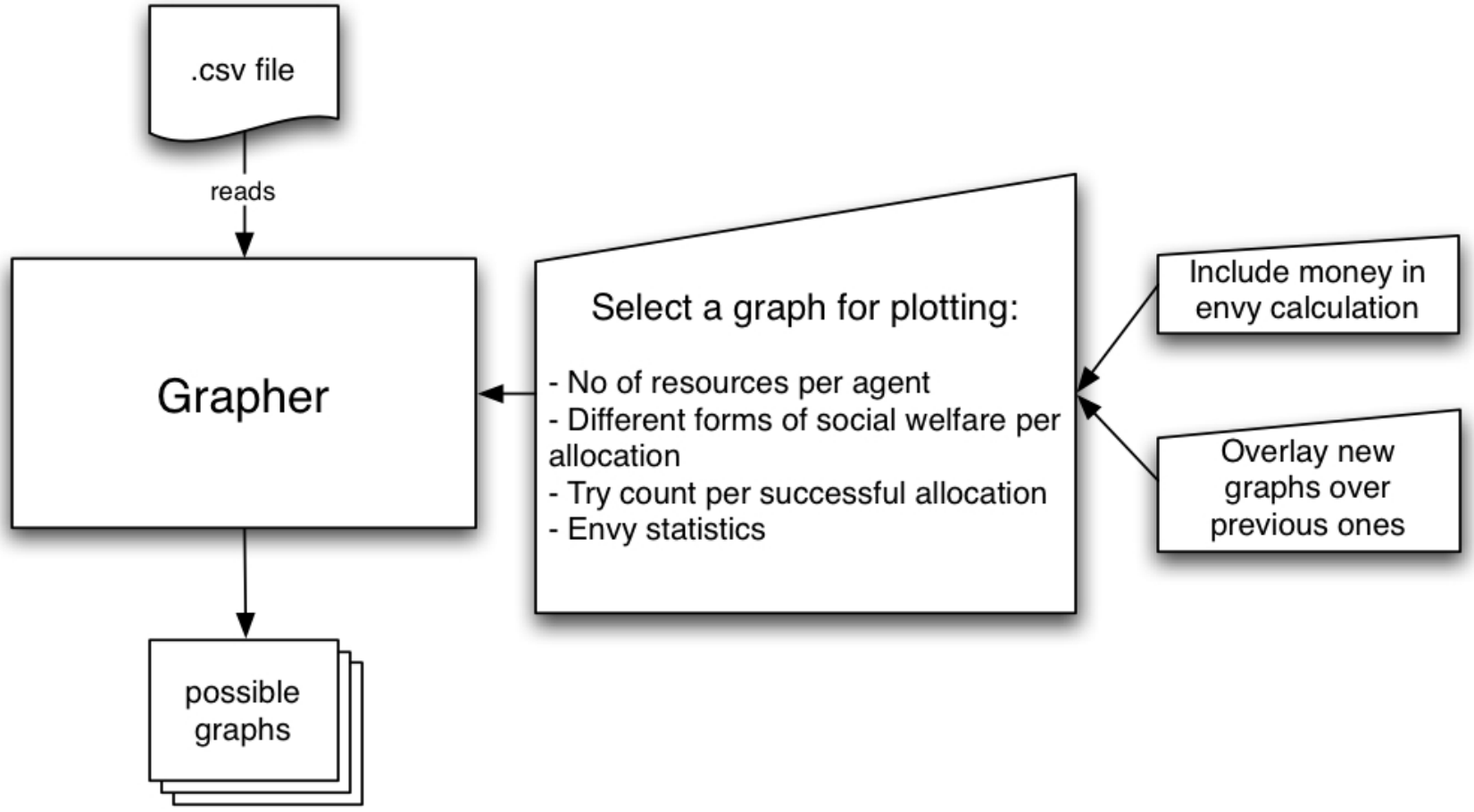
bilateral deals

- Randomly select a pair of agents
- Optimally redistribute all their resources by means of optimal partial reallocation.

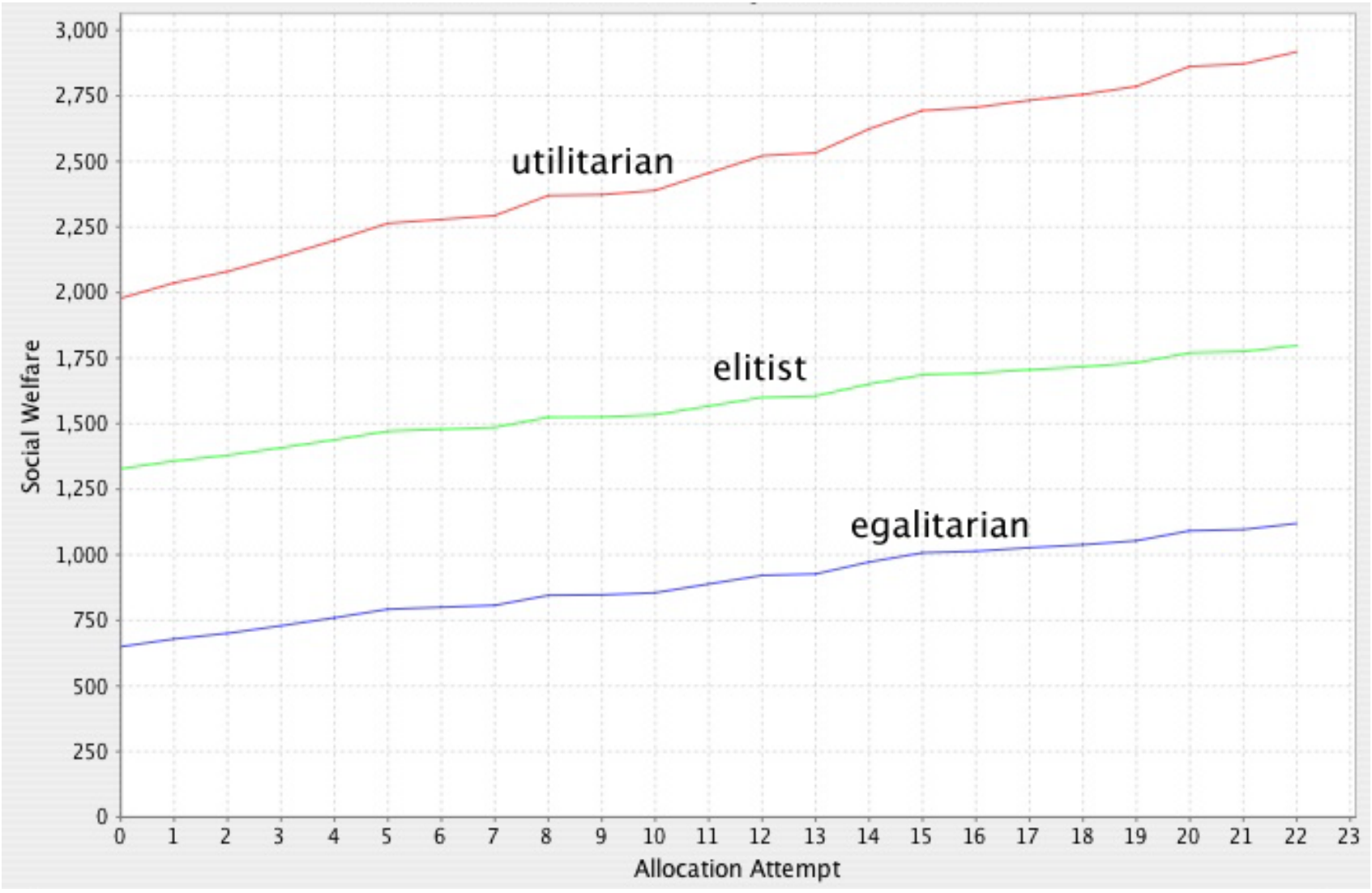

.sdf file
reads
ExperimentRunner
outputs
.csv file
Select a payment function
Select a deal finding type
Select number of allocation tries

MADRAS: Running an Experiment

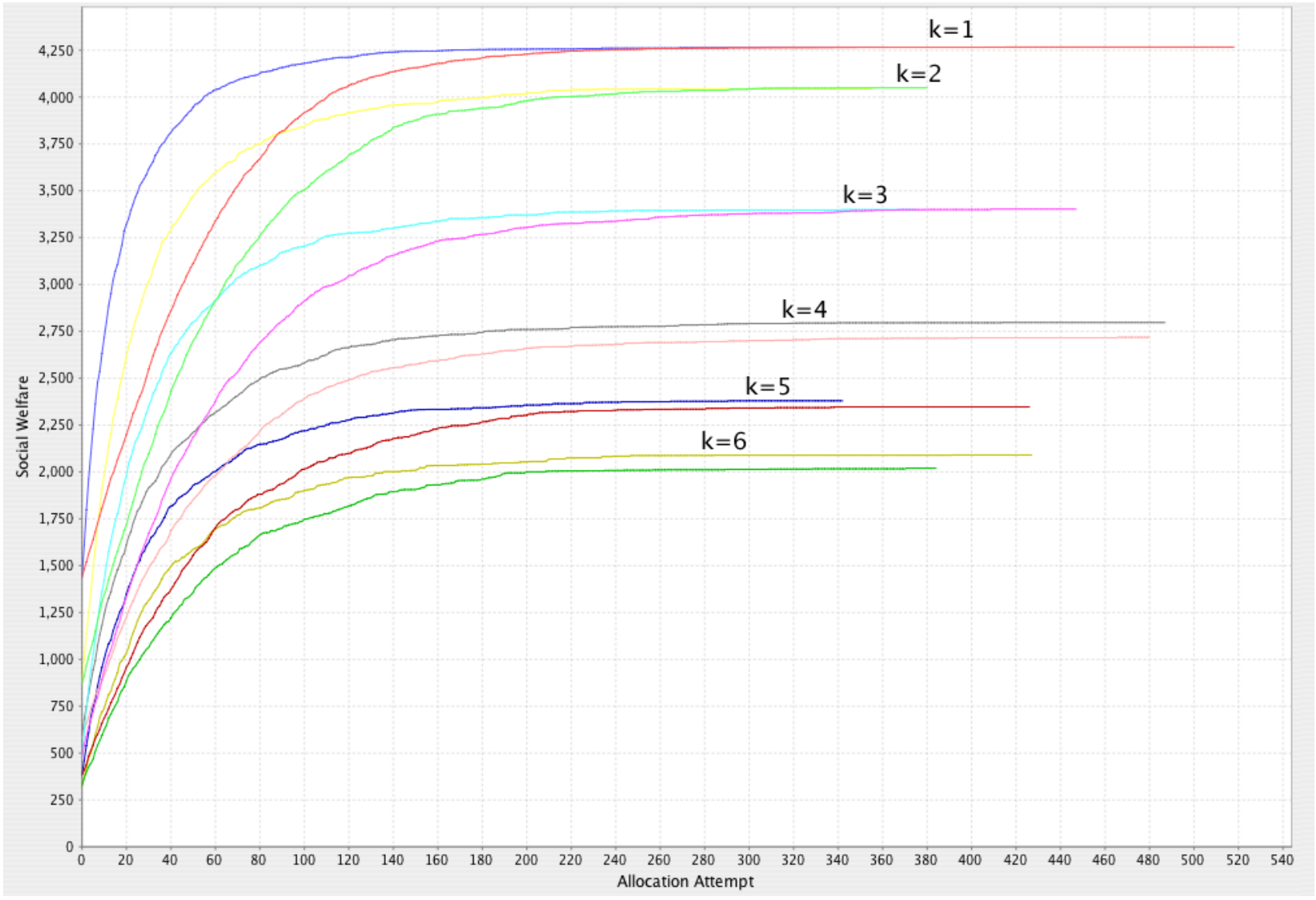



.csv file
reads
Grapher
possible graphs
Select a graph for plotting:
- No of resources per agent
- Different forms of social welfare per allocation
- Try count per successful allocation
- Envy statistics
Include money in envy calculation
Overlay new graphs over previous ones

Example with 2 agents



Utilitarian SW for 30 goals per agent, with a variable *k*



Conclusion

The distributed approach to MARA is attractive: it allows to share computational load and requires no central control.

But understanding the dynamics of distributed MARA is difficult. Simulation using MADRAS can help testing hypotheses.

MADRAS consists of three independent modules:

- Scenario Generation
- Experiment Running
- Visualization of Results

Future work:

- Valuation generation: other types + more realistic
- Calculate optimal allocation (for comparison)
- Different agent rationalities

Get a copy of MADRAS at http://madras.infosyncratic.nl