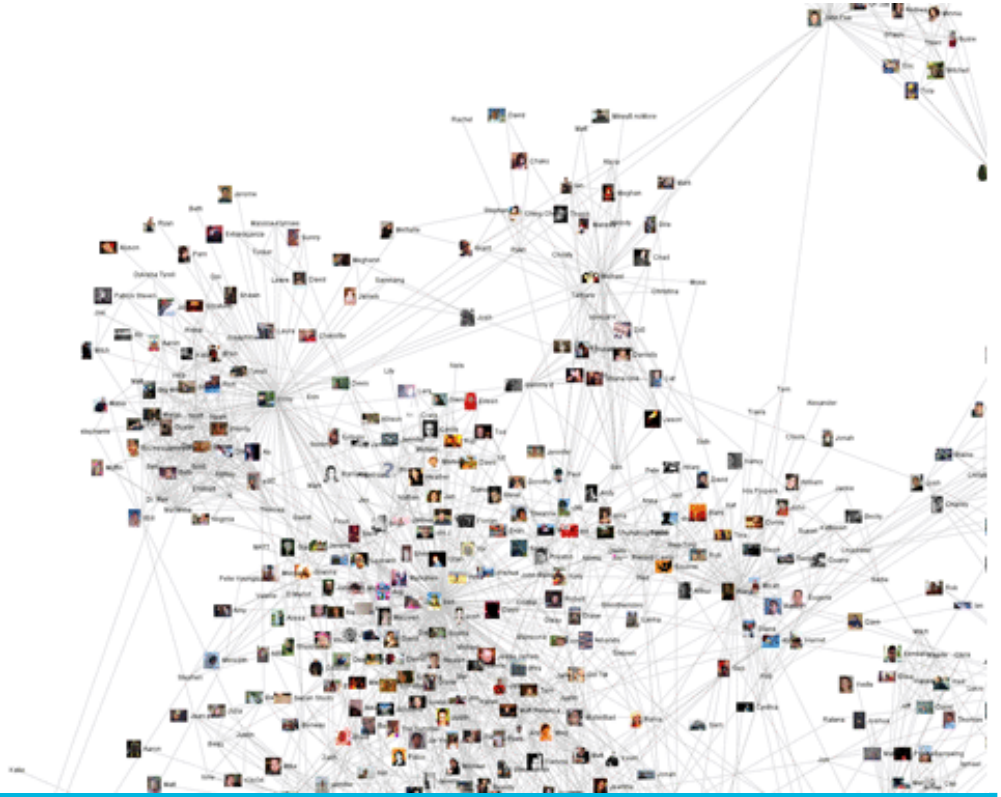
Resource Allocation in Social Networks



Mathijs de Weerdt
Yingqian Zhang, Tomas Klos

June 4, 2008

**Faculty of Electrical Engineering, Mathematics and Computer Science,
Department of Software Technology**

CWI, Amsterdam, SEN-4

TU Delft
Delft University of Technology

Ideas in this talk

1. Variant of resource allocation problem, i.e., in a social network

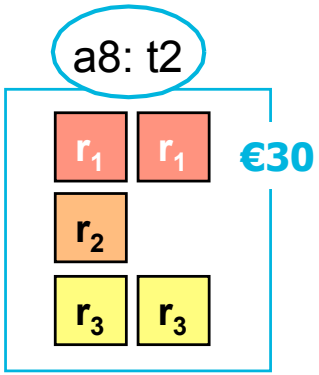
2. Dealing with resources as private information of strategic agents

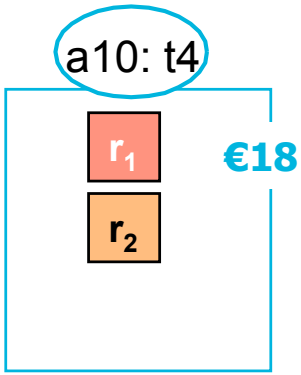
Overview

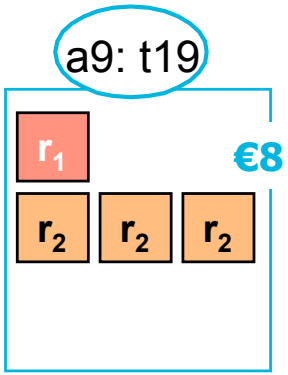
- Problem Definition
- A Greedy Distributed Protocol
  - Algorithm
  - Run-time analysis
  - Experiments
- Mechanism Design
  - Optimal + VCG
  - Greedy mechanism
  - Another Payment Function
- Conclusions & Future Work

Resource allocation

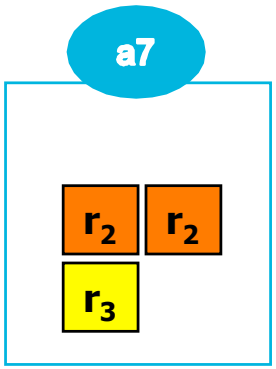
Agents value certain resource combinations (eg execute tasks)

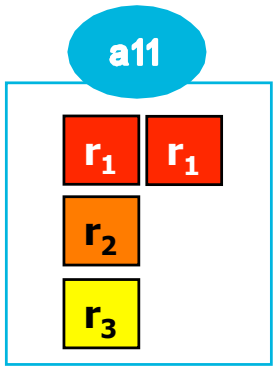


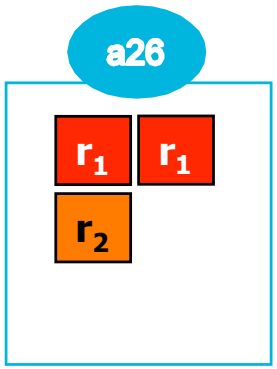




Resources initially reside with other agents

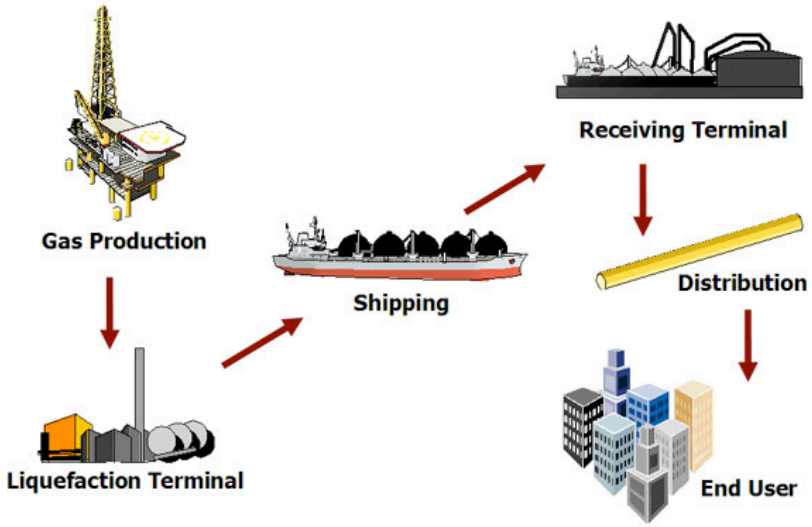






TU Delft

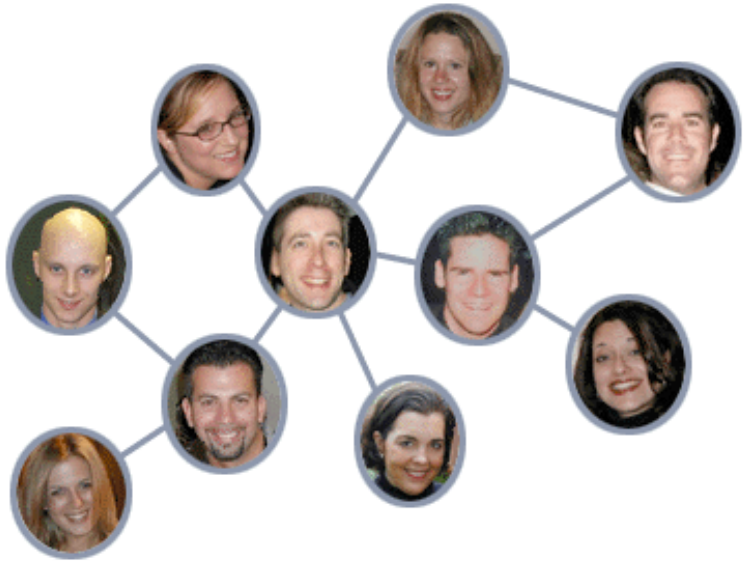
Why social networks?



Social relations important in real-world task allocation:

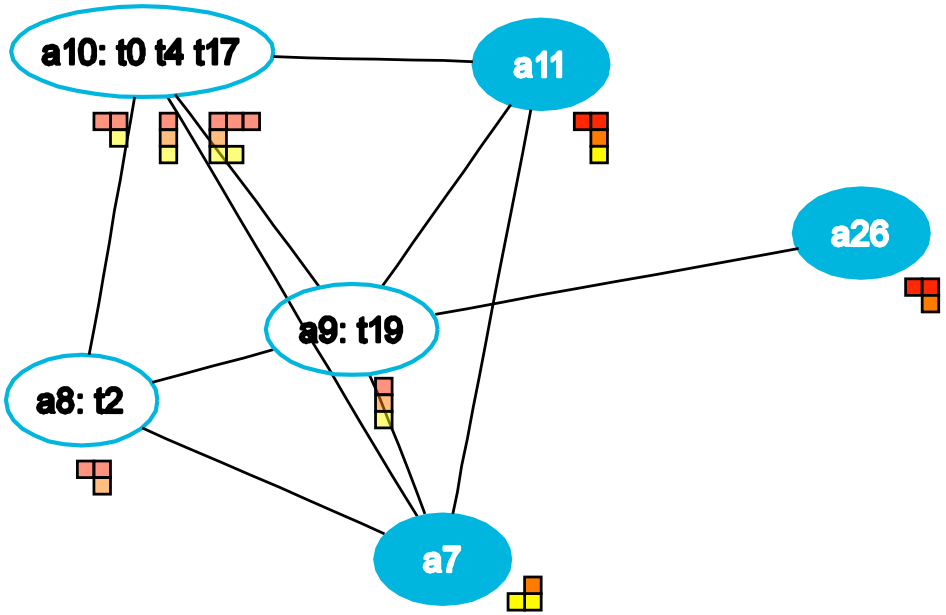
- Industrial procurement, eg supply chain formation
- Free-lancers networks

→ *preferred partnerships* instead of plain markets

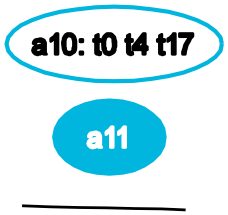


TU Delft

Resources in Social Networks



- Each agent has:
  - resources
  - tasks with utility
  - connections
- Each task $t \in T$
  - requires resources *rsc(t)*
  - has a utility *u(t)*



agent a10$\in A$ with three tasks$\in T$ (manager)

agent a11$\in A$ without tasks (contractor)

connections between two agents: allowed to allocate/cooperate

TU Delft

Problem Definition: Resource Allocation in a Social Network

- Given
  - a network of potential partners, where
  - some agents have resources
  - other agents have tasks, and thus utilities for combinations of resources,
- determine a resource allocation (to neighbors) such that sum of utilities (of fully satisfied tasks) is maximal.

Greedy distributed protocol (GDAP)

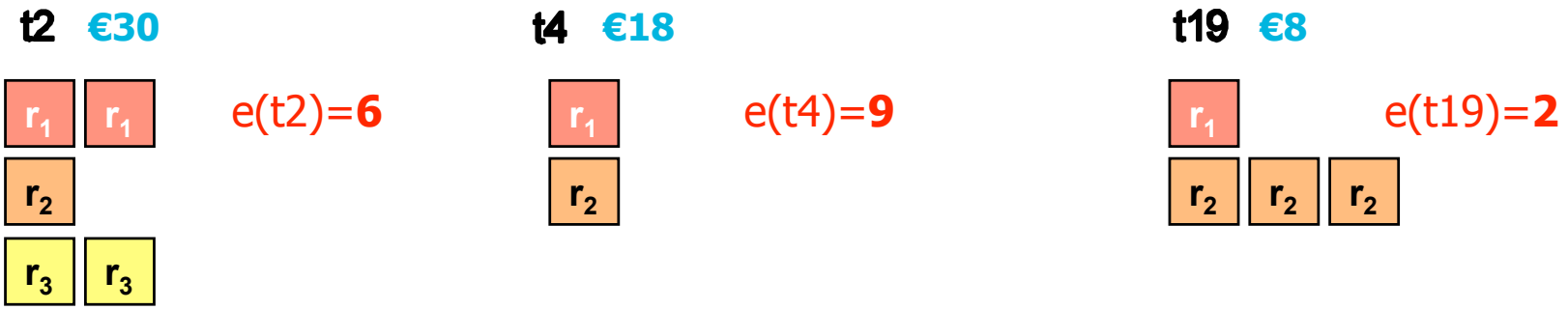
Idea

First allocate resources to tasks that have high utility and require few resources

Definition

The efficiency e(t) of a task t is:

$$e(t) = \frac{u(t)}{\sum_{r \in R} rsc(t,r)}$$



TU Delft

Greedy distributed protocol (GDAP)

m (manager): agent that has utility (task) for a combination of resources of different types

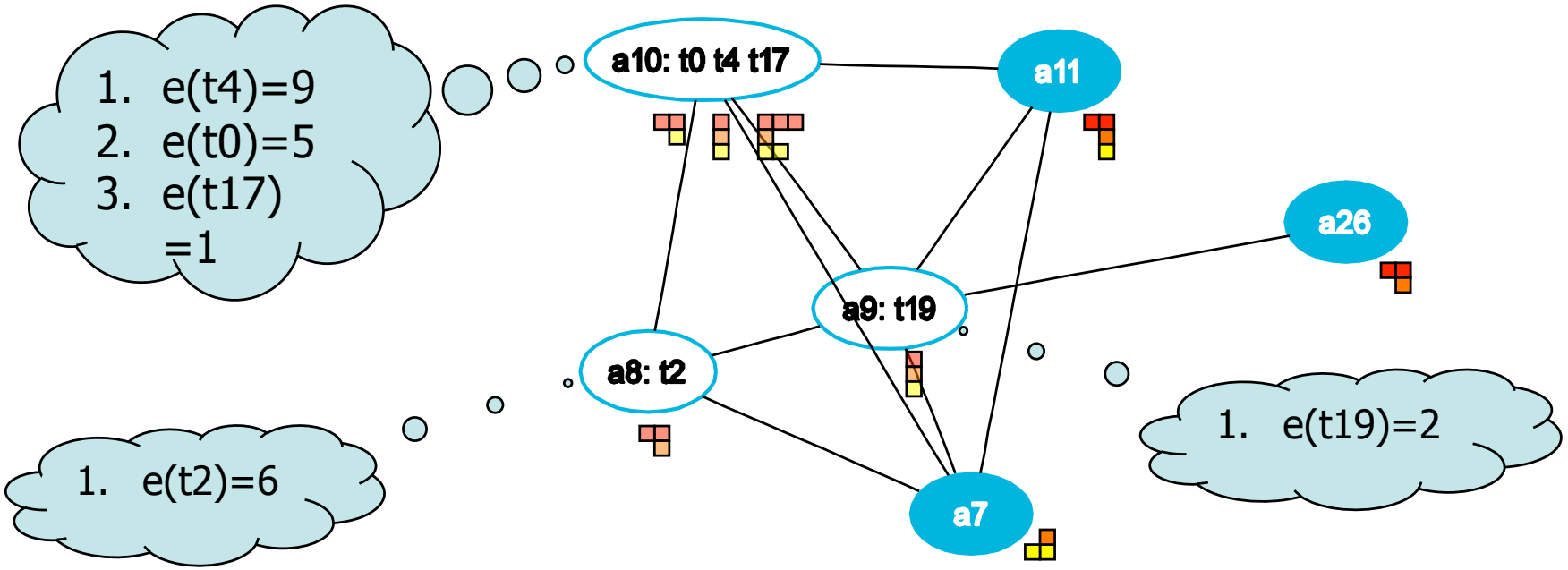
c (contractor): agent that can provide a number of resources

Repeat

1. **m**: Send requests for resources for most efficient task to *neighbors.*
2. **c**: Offers resources to request with *highest* efficiency.
3. **m**: If task can be fully allocated, do so and remove it.
4. **m**: Else, if all neighbors offered, remove it

Until no tasks are left

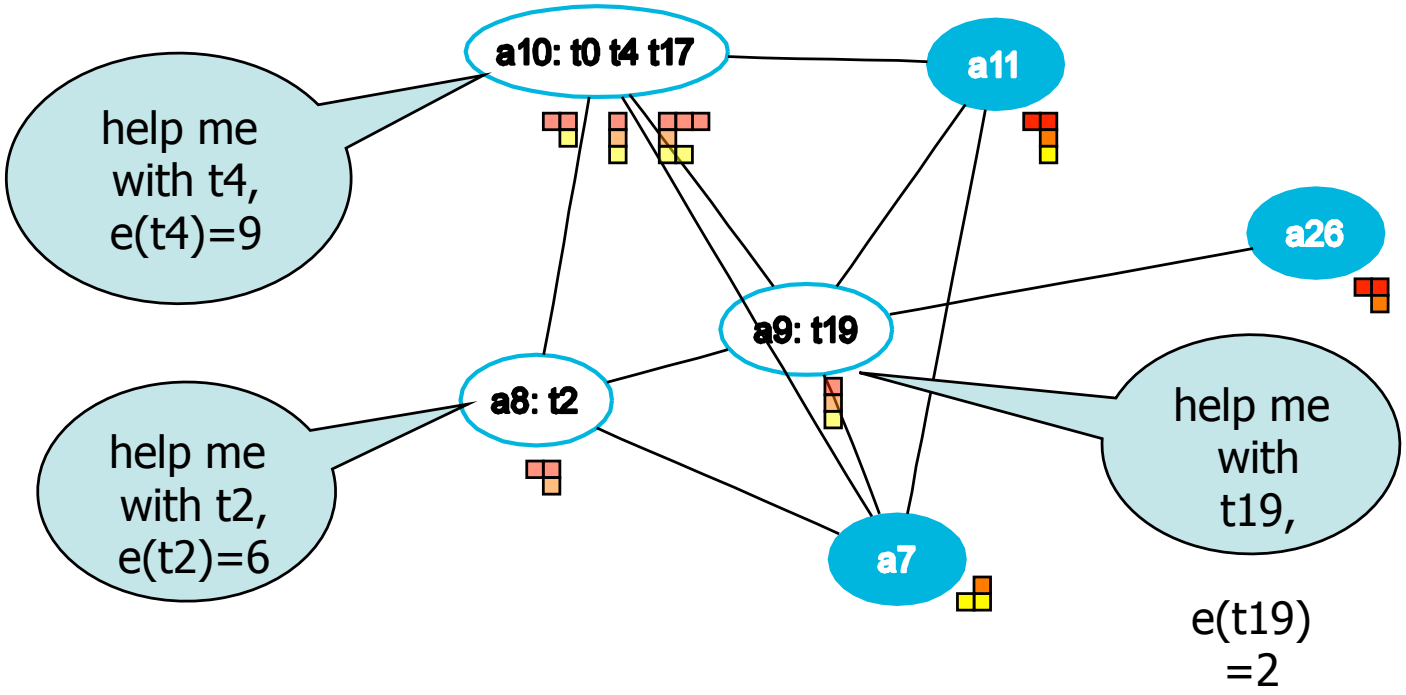
Greedy distributed protocol (GDAP)



1. **m**: Each manager agent calculates the *efficiency* $e(t)$ for its tasks T_a; sorts tasks in descending order of efficiency: $e(t) = \frac{u(t)}{\sum_{r \in R} rsc(t,r)}$

TU Delft

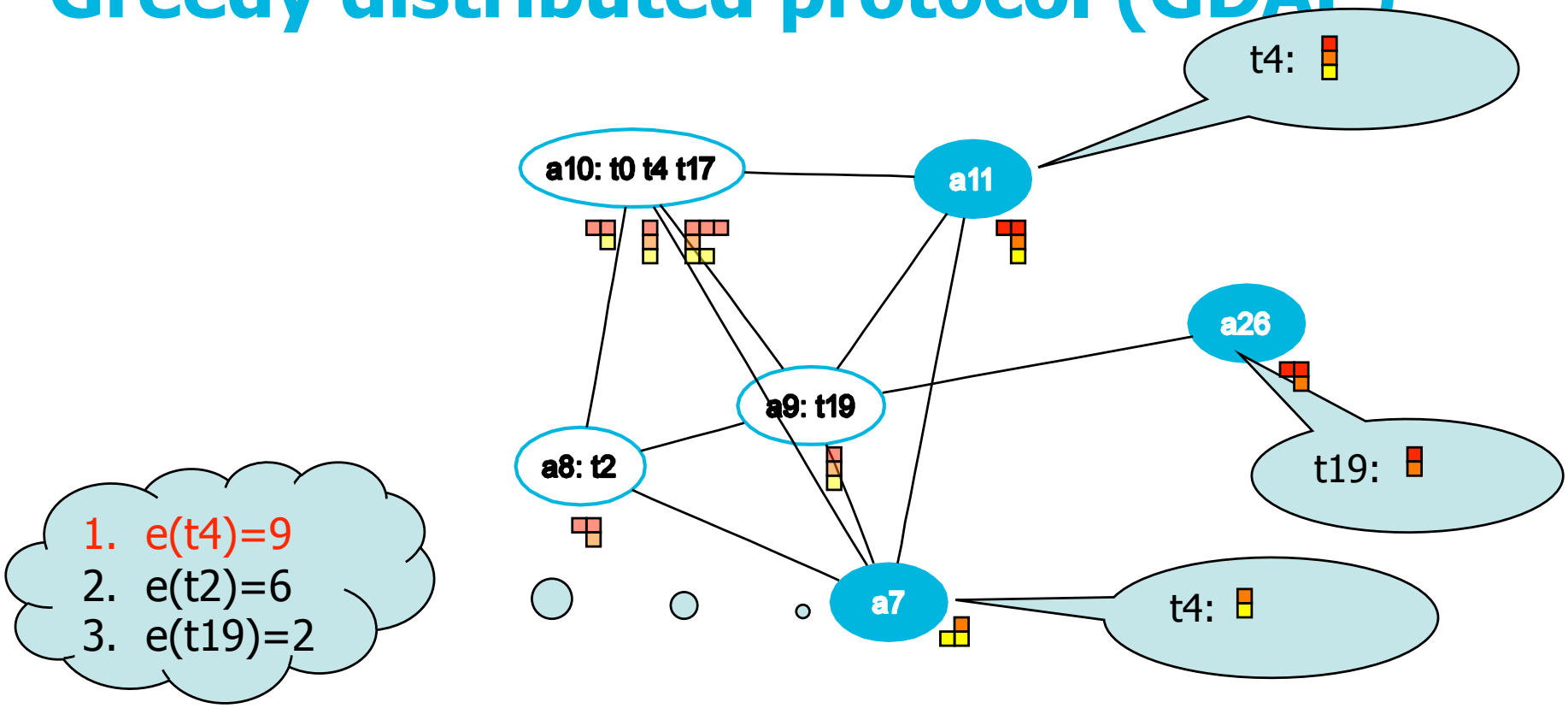
Greedy distributed protocol (GDAP)



1. **m**: Send requests for resources for most efficient task to *neighbors.*

TU Delft

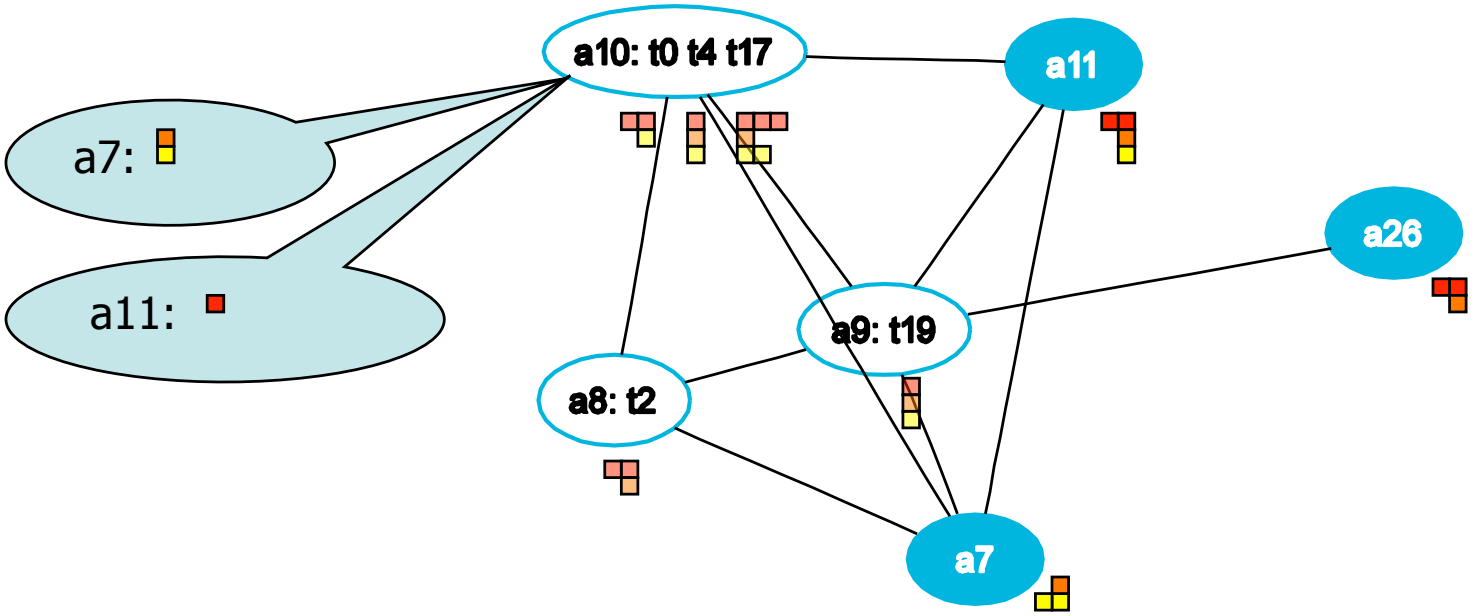
Greedy distributed protocol (GDAP)



2. c: Offers resources to request with *highest* efficiency.

TU Delft

Greedy distributed protocol (GDAP)



3. **m**: If task can be fully allocated, do so and remove it.
4. **m**: Else, if all neighbors offered, remove it.

TU Delft

Run-time analysis

For a social resource allocation problem with n tasks and m agents

- O(n) iterations
- per iteration: O(m) operations (in parallel)

- so the run-time of GDAP is O(nm).
- The number of communications messages is
  - per iteration (n), per task (n), O(m)
- so number of communication messages is $O(n^2m)$.

Experiments

- Objective: study performance of the greedy distributed algorithm GDAP in different problem settings:
  - Network topology / degree
  - Resource ratio: (# resources req'd)/(# resources available)

Measurements

- *Computation time*
- *Solution quality* (utility of tasks allocated)
  - for small problems: GDAP/OPT
  - for large problems: GDAP/Upper Bound

Experiments (OPT)

- OPT: by translation to ILP:

$$\texttt{Maximize} \quad \sum_{j=1}^{n} y_j \cdot U(t_j)$$

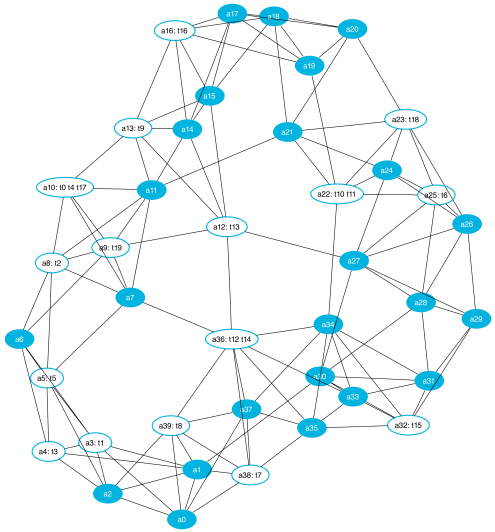
$$\forall_{1 \leq j \leq n} \forall_{1 \leq k \leq l} \quad \sum_{\{i \in [1,m] | (i, loc(t_j)) \in AE\}} x_{ijk} \geq y_j \cdot req(t_j)(r_k),$$

$$\forall_{1 \leq i \leq m} \forall_{1 \leq k \leq l} \quad \sum_{j=1}^{n} x_{ijk} \leq rsc(i)(r_k).$$

Experiments (Upper bound)

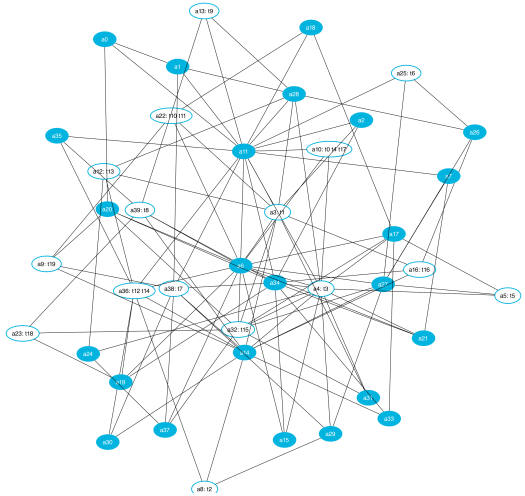
- Assume divisible goods
- Represent as min-cost network flow problem:
  - node *a* for every agent-available-resourcetype > 0
    - edge from *s* to *a* with this as capacity
  - node *b* for every task-requested-resourcetype
    - edge to *t* with this as capacity and cost: -efficiency
  - edge from *a* to *b* if agents are neighbors

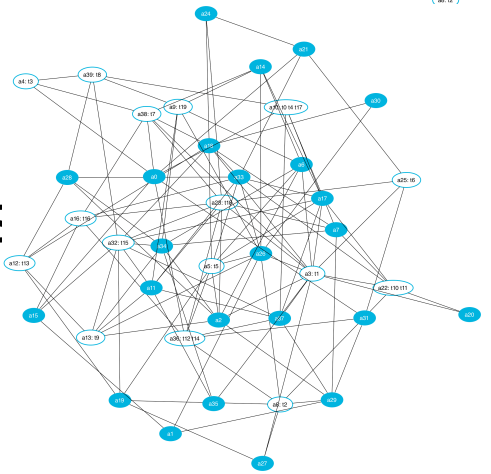
Experimental settings



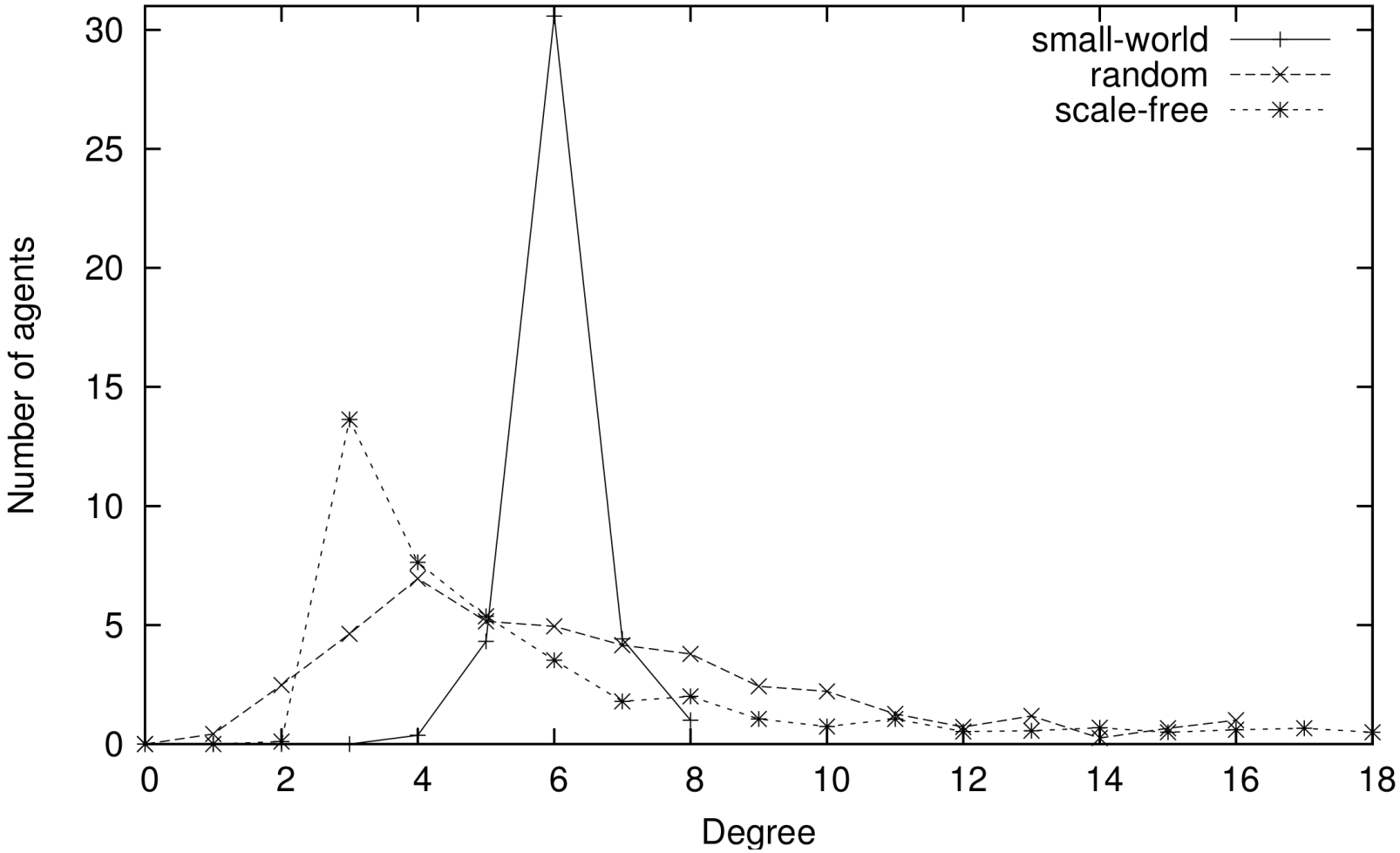
Social network structures

- *Small-world network* (Watts, Strogatz, 1998): average shortest path length scales O(log n), even with few long links
- *Scale-free network* (Barabasi, Albert, 1999): few agents have many neighbors; many have only a small number of neighbors
- *Random network* (uniform): agents are randomly connected





Degree histogram



TU Delft

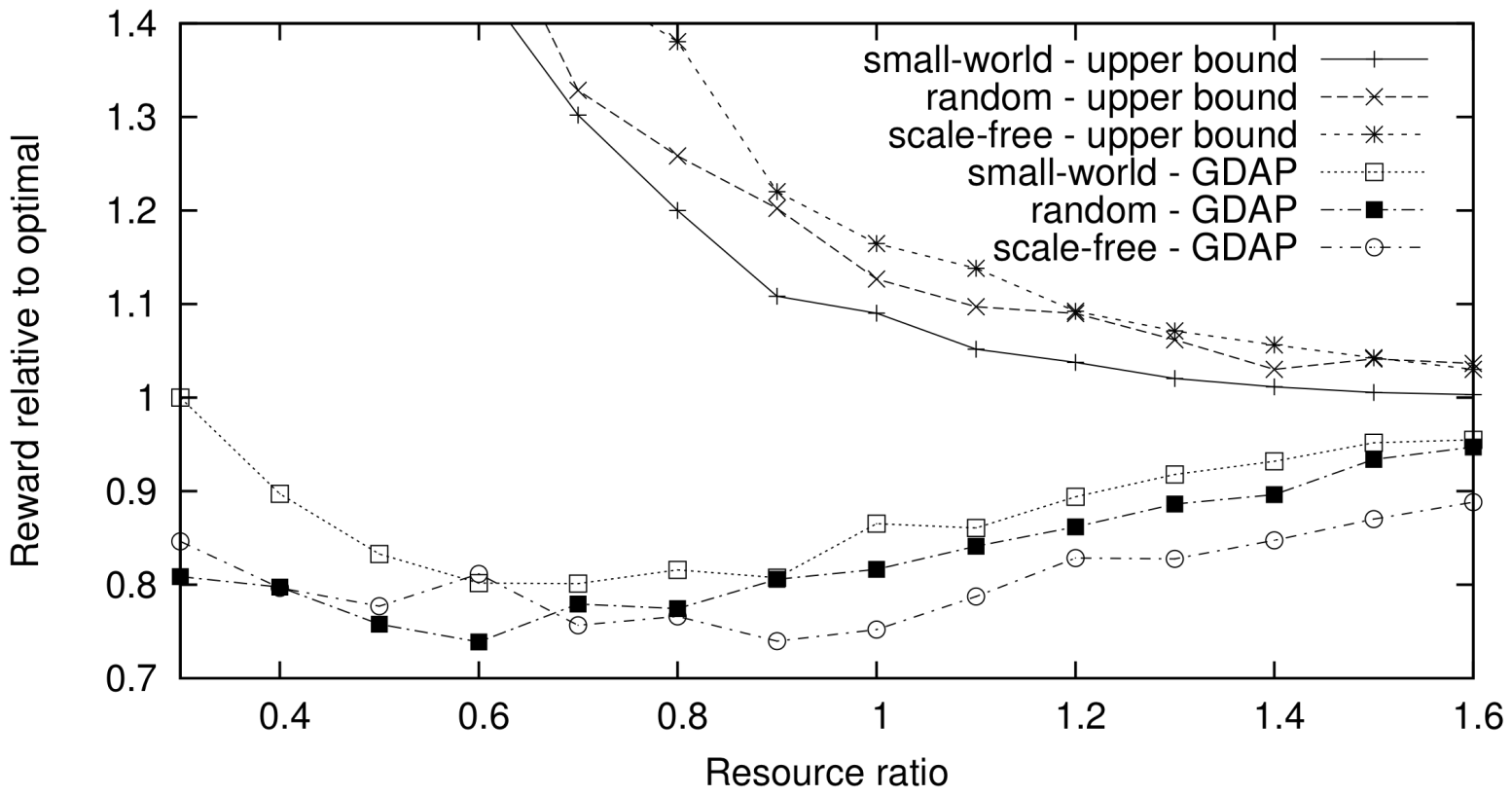
Experiments

- Objective: study performance of the greedy distributed algorithm GDAP in different problem settings:
  - Network topology / degree
  - Resource ratio: (# resources req'd)/(# resources available)

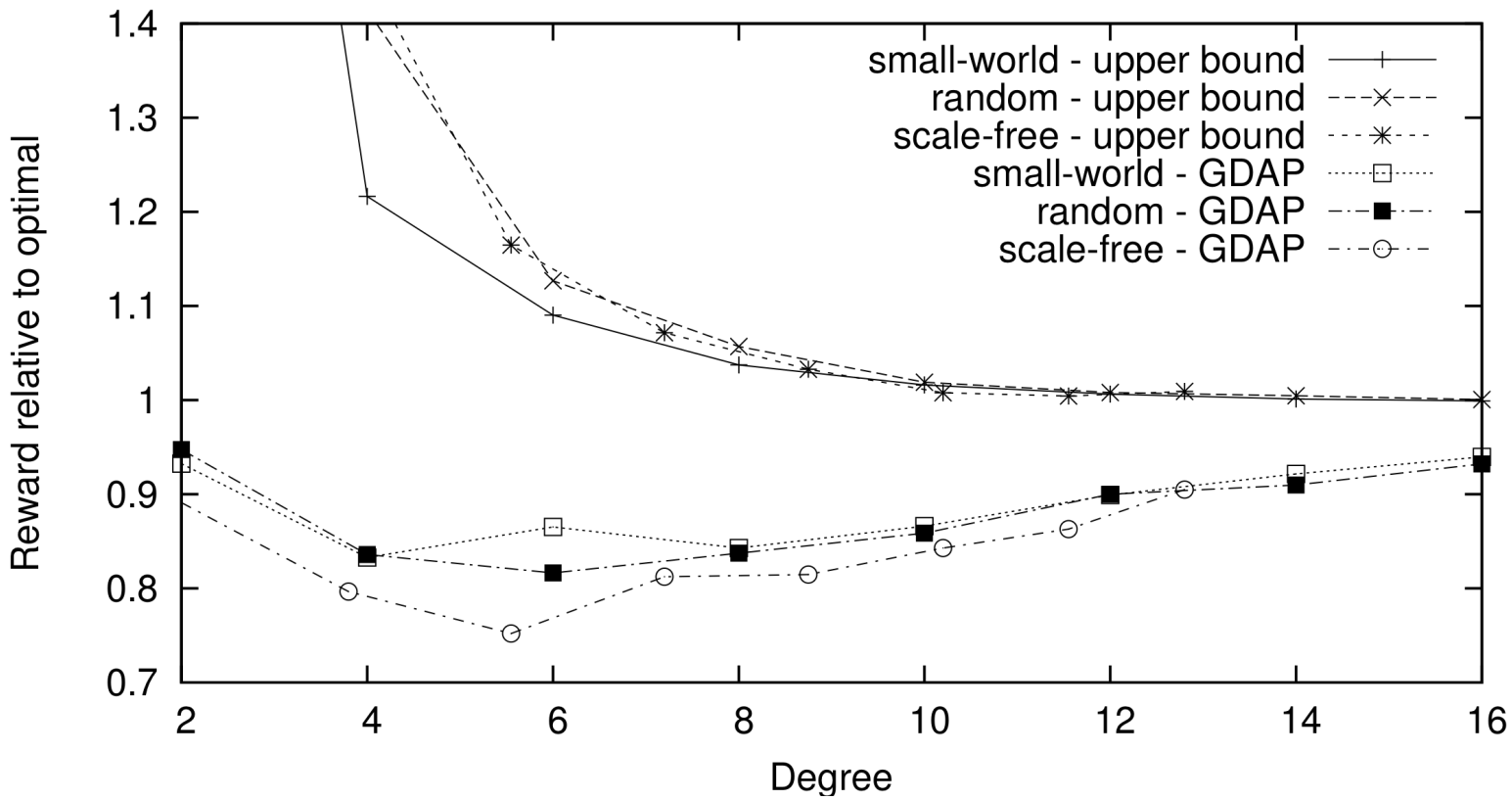
Measurements

- *Computation time*
- *Solution quality* (utility of tasks allocated)
  - for small problems: GDAP/OPT
  - for large problems: GDAP/Upper Bound

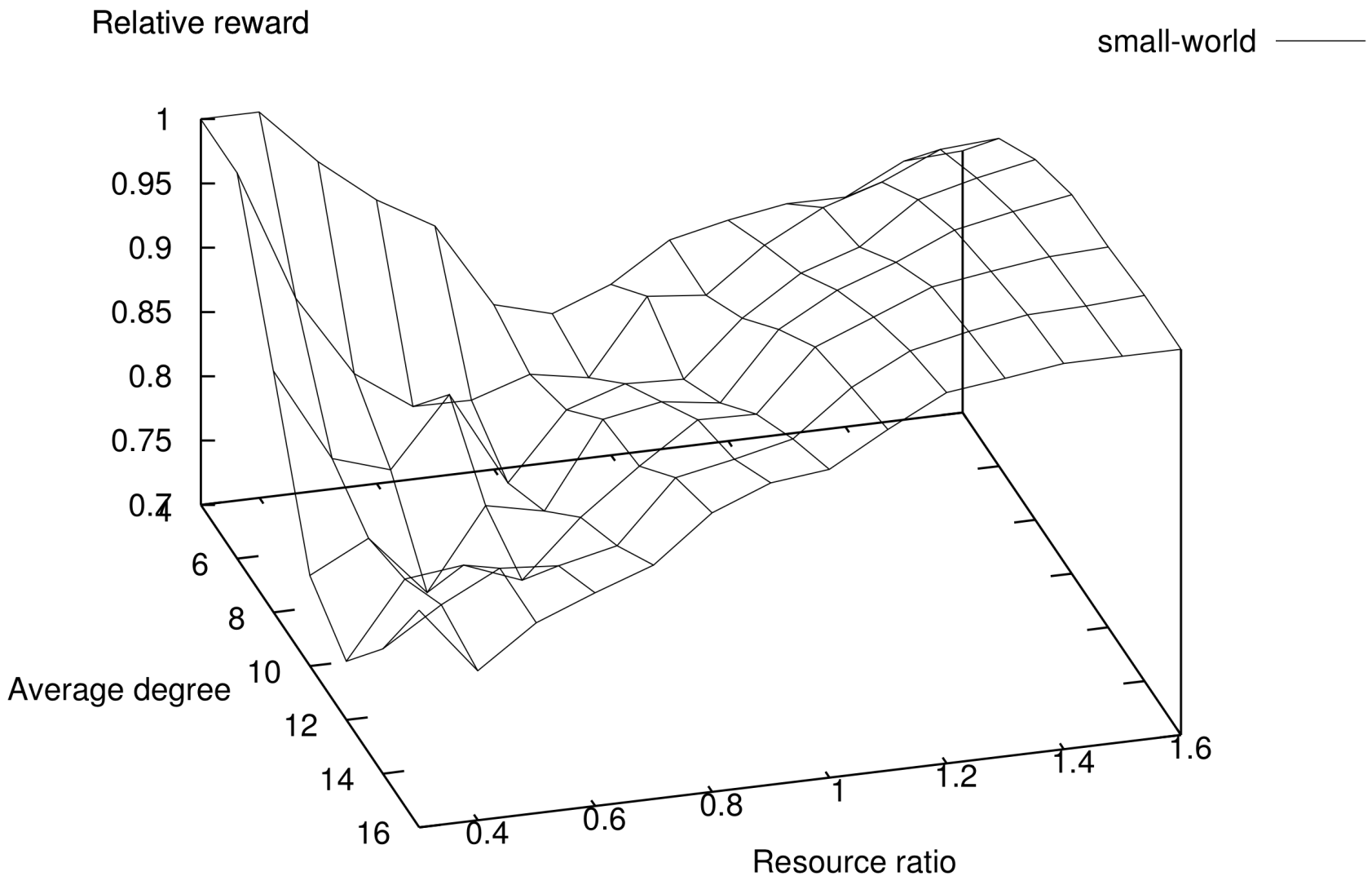
Setting 1a: 40 agents, 20 tasks, average network degree 6, uniform task utilities, varying resource ratio (total available resource / total required resource)



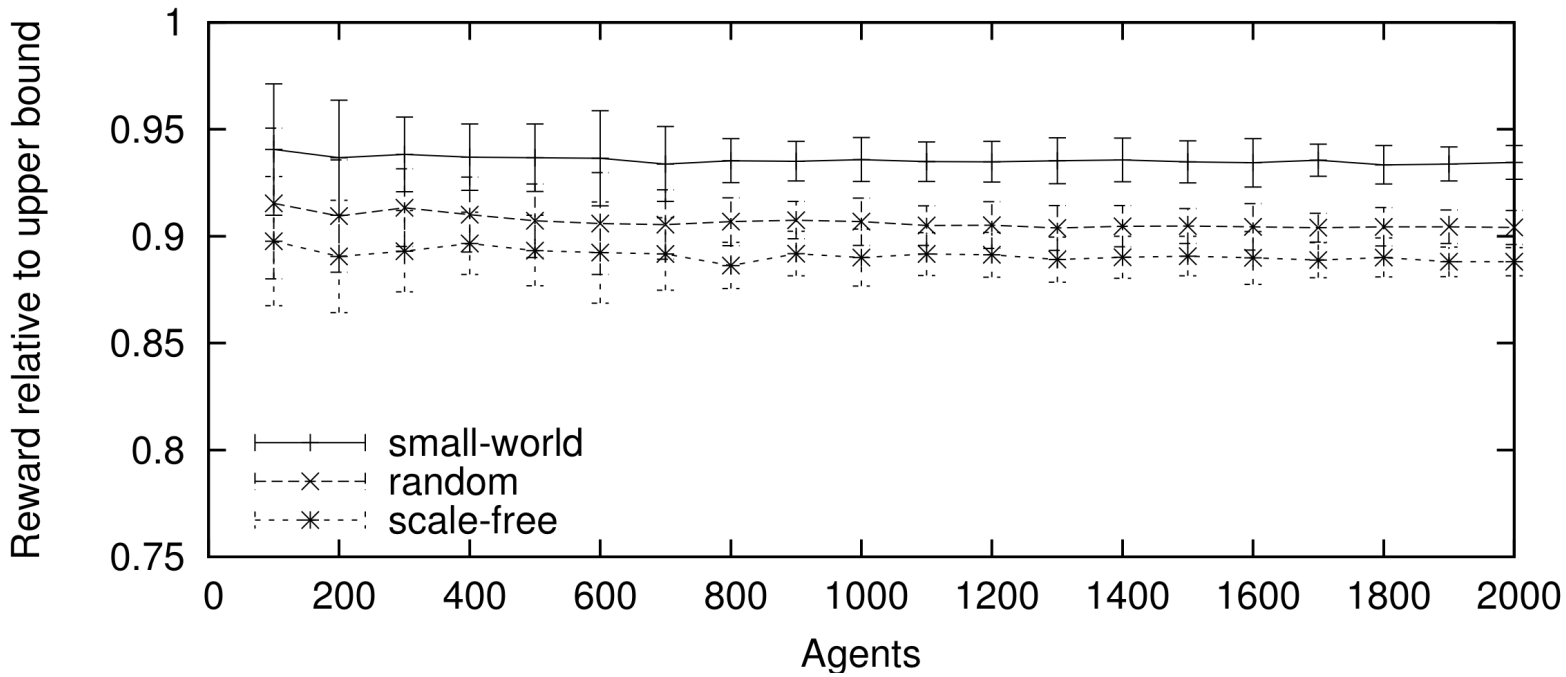
Setting 1b: 40 agents, 20 tasks, uniform task utilities, resource ratio 1.2, varying degree



Setting 1 overall: 40 agents, 20 tasks, uniform task utilities, varying both resource ratio and degree



Setting 3: resource ratio 1.2, degree 6, size ratio of agents and tasks 5/3, varying number of agents from 100 to 2000.



Summary of results

- GDAP performs well (around 90%) when there are sufficient resource available
  - high resource ratio,
  - and/or high degree
- performs around 70% when resources are scarce
- slightly better on small-world networks
- very fast (computation time less than 2s for 2000 agents)

Mechanism Design

- Two different agents
  - *Contractor* agents are *self-interested*, maximizing utility $u_i(o)$; in this setting basically the *payment*
  - Task manager agents are cooperative
- Public information:
  - social network
  - task information: location; utility
- Private information:
  - contractor agents' available resources
- Goal: a mechanism that is
  - incentive compatible for contractor agents
  - efficiently computable
  - as good as possible

Exact mechanism with VCG payment

- Exact mechanism OPT by transformation to ILP
- VCG payment: marginal utility to social welfare

  $p_i = v_i(o) + W(o) - W(o_{-i})$

Properties

- incentive compatible with respect to under-reporting
- over-reporting may lead to infeasible outcomes
- exponential algorithm
- optimal outcome

Greedy mechanism with VCG payment

- order tasks on efficiency (value/#resources)
- T = $\varnothing$
- for each task *t*
  - check using network flow if adding *t* to T is feasible
  - if so add *t* to T, otherwise delete *t*

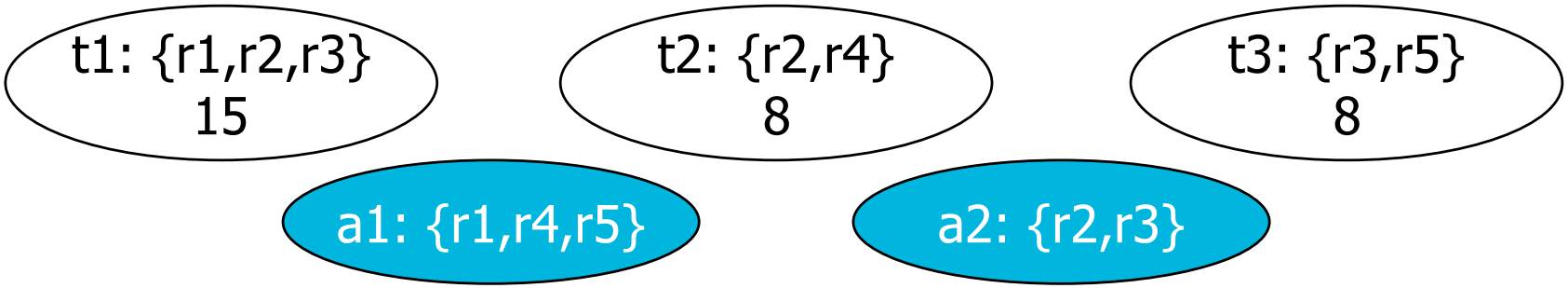
Properties

- polynomial algorithm, #resources-approximation
- VCG payments cannot make Greedy incentive compatible (with respect to under-reporting)…

VCG and approximations

Theorem: VCG payments cannot make Greedy incentive compatible (with respect to under-reporting)

- a1 is better off reporting r4 and r5 (payment 16) than reporting also r1 (payment 15)



- in line with Nisan & Ronen (00/07) result on combinatorial auctions (reasonable & not optimal -> VCG not truthful)

Greedy mechanism with alternative payment

- Greedy payment:
  - order all tasks on efficiency (value/#resources)
  - for *each* task t
    - pay all agents that sell essential resources (to t)
    - delete those resources

Properties

- Greedy mechanism is incentive compatible wrt under-reporting
- because payment monotonically increasing in declared resources
- -W(o) ≤ total payment to contractors ≤ U(T)

Preventing over-reporting

- Deposit mechanisms:
  - first ask each agent to pay sum of task utilities as deposit
  - calculate solution
  - if agent delivers promised resources, return deposit

Contributions

- Problem: resource allocation in social network setting
- efficient distributed protocol
- VCG cannot prevent over-reporting (leading to infeasible outcomes) even with OPT
- VCG does not prevent under-reporting with a Greedy (non-optimal) algorithm either, while
- a "Greedy" payment can prevent under-reporting (budget-imbalance depends on social network setting)
- over-reporting can be prevented by asking a deposit

Future Work

- mechanism where manager agents may also strategize
- budget balance:
  - search for (weakly) budget balanced payment, or
  - prove non-existence and analyze experimentally
  - give also better bound on deposit
- online mechanism: tasks and resources arrive over time
- distributed mechanism: only local payments

References

- Mathijs M. de Weerdt and Yingqian Zhang and Tomas B. Klos (2007). Distributed Task Allocation in Social Networks. In Michael Huhns and Onn Shehory (Eds.). *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)*, pp. 488--495. Research Publishing Services.
- Yingqian Zhang and Mathijs M. de Weerdt (2007). VCG-based Truthful Mechanisms for Social Task Allocation. In *Proceedings of the Fifth European Workshop on Multi-Agent Systems (EUMAS-07)*, pp. 378--394.
- Mathijs M. de Weerdt and Yingqian Zhang (2008). Preventing Under-Reporting in Social Task Allocation. In Han La Poutre and Onn Shehory (Eds.). *Proceedings of the 10th workshop on Agent-Mediated Electronic Commerce (AMEC-X)*.
- Roman van der Krogt and Mathijs M. de Weerdt and Yingqian Zhang (2008). Of Mechanism Design and Multiagent Planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-08)*.

TU Delft