Distributed Optimization in Networks

Work-in-progress report

Satu Elisa Schaeffer

Laboratory for Theoretical Computer Science, Tkk

elisa.schaeffer@tkk.fi

Rutgers-HeCSE workshop, May 9, 2006

OUTLINE

- Motivation for distrib. optimization
- Ad hoc networks
- Sensor networks
- Peer-to-peer systems
- Throughput optimization

OPTIMIZATION

$=$ the task of making some **selections** such that an *objective function* reaches the **best possible** value while respecting a set of *constraints*

A *feasible* solution $=$ a selection that **fulfills** all the constraints

Typical examples: the maximization of profits and the minimization of costs or damages

Some network optimization tasks

- finding the **maximum flow** from a source node to a target node with respect to edge capacity constraints
- finding the **vertex cover** of *minimum* order such that each vertex is either included in the cover set or has a neighbor that is in the cover set
- finding a **coloring** for the graph that *minimizes* the number of colors needed, when no two neighboring vertices may share a color

Motivation for distributed optimization

Optimization under circumstances where *not all* information is globally available or readily accessed simultaneously.

The **correctness** of distributed algorithms is not trivially deduced.

General model of computation

- a set of independent agents
- **goal**: global optimum using only local information
- each agent sets a single primal variable knowing *only* the constraints affecting that variable
- communication by fixed-size messages between immediate neighbors

Approximation algorithms

$(1+\epsilon)$-approximation to the optimum of a positive LP with a polylogarithmic number of local communication rounds [BBR04]

primal and dual feasibility: iteration *pairs* (violating a dual constraint to fix primal feasibility and then moving back to fix feasibility for the dual)

AD HOC NETWORKS

- self-organizing, dynamical networks
- nodes join and depart independently
- network nodes may be stationary and/or mobile (MANET)

SPANNING TREE CONSTRUCTION

- minimum-diameter, degree-limited spanning tree (**NP**-hard)
- usable e.g. in overlay multicast
- approximate optimization by local adaptations [CCK04]
- adapting to changes in network topology
- stress $=$ number of identical packets per link
- stretch $=$ ratio of path length on the tree to node distance

SENSOR NETWORKS

$=$ collections of *sensor nodes* spread around an area in which a certain phenomenon of interest is expected to take place

In many cases, the sensor placement is *not* a carefully designed process but more of a random scattering.

SENSOR COMPONENTS

- a *sensing* unit that makes observations of the environment
- a *processing* unit that determines what actions need to be taken (a limited computational device with little memory)
- a *transceiver* unit that receives and broadcasts signals enabling nearby sensor nodes to communicate; usually the range of the broadcast is somewhat limited
- a *power* unit (essentially a battery) that supplies energy for the other components; the battery life of the nodes governs the life-time of the network

Energy-efficient routing

In radio-communication networks, *routing* of the network traffic should be done efficiently w.r.t. the time and energy used.

Additional problems caused by interference of broadcasts, broadcast storms, and other curious effects in (wireless) message propagation.

NETWORK LIFETIME OPTIMIZATION

A basic setup

- stationary wireless sensor nodes with limited energy
- each sensor may adjust its transmission power
- ignoring bandwidth and interference limitations
- **goal**: *maximize the network lifetime* (instead of simply minimizing the total energy consumption)

Peer-to-peer (P2P) systems

distributed systems composed of **independent** computers that work together to achieve a common goal, usually involving the **sharing** of computing, file or network resources

P2P ARCHITECTURES

1. networks with **centralized topology**, content information and structural (e.g. the original Napster, based on a full directory of peers)
2. decentralized but **structured networks** (e.g. Freenet): topology is imposed in a central manner, but the functions is decentralized
3. decentralized and unstructured networks, such as Gnutella

Determining the coordinates

- **setup**: network formed by scattered sensors
- each sensor is capable of measuring the distances to its closest neighbors
- global coordinates *unknown*
- **goal**: construct a “realistic” coordinate system [GK05]
- **why?** — allows for efficient geographic routing

FORMULATION

- **input**: graph G, edge lengths ℓ_{ij}
- **task**: find an optimal layout $\boldsymbol{p}$, where $p_i \in \mathbb{R}^2$ is the location of sensor i s.t. $\forall j \neq i$

$$\begin{cases} ||p_i - p_j|| = \ell_{ij}, & \text{if } (i,j) \in E, \\ ||p_i - p_j|| > \max\limits_{(i,j)\in E} \ell_{ij}, & \text{otherwise.} \end{cases}$$

- minimizing a localized *stress function* that (SSQ of $d_{ij} - \ell_{ij}$)
- initial layout influences the outcome
- iterations: solving LPs in a distributed fashion

GAME THEORY AND P2P SYSTEMS

- “incentive to share” [GLBML01] (free-riding problem)
- trust, access, bandwidth, ...
- similar issues arise in ad hoc networks
- mechanism-design approaches [SP03]

ROUTING AS A COALITION GAME

- routing $\approx$ a *multicommodity* flow task
- $\Rightarrow$ a coalition game
- the game has a non-empty core [MS05]

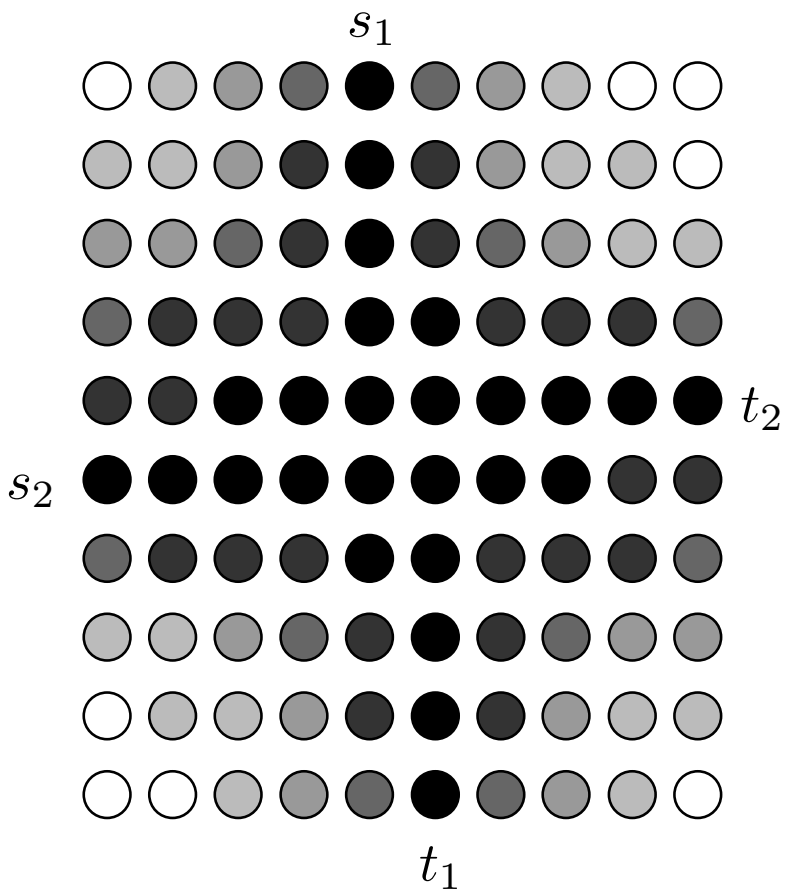
THROUGHPUT MAXIMIZATION

Case study: a simplified problem of maximizing the throughput of a communication network with multiple source-destination pairs

- theoretically equivalent to a multicommodity flow problem
- has a formulation as a coalition game with a non-empty core [MS05]

Setup for the case study

- two source-destination pairs communicate on a **steady bit rate** over a **grid topology**
- the traffic pattern resembles sending a live-video stream from a server to a client
- we *do not* consider energy-limitations explicitly
- we aim for **high throughput**, which is likely to improve the network lifetime


s_1
t_2
s_2
t_1

BASELINE IMPLEMENTATION: DSR

- *Dynamic Source Routing protocol* [JMB01, JMH04]
- chooses a path between the source and destination nodes from its cache and routes all traffic along this path as long as the path is operational
- route information embedded on the data packet
- route discovery messages are triggered periodically with exponential back-off (to prevent flooding)

ROUTE-REQUEST PACKETS

- contain a **source node identifier** and a **unique packet identifier** that allows intermediate nodes to only forward each route request once
- built by the forwarding nodes that **append** to the packet their own information
- the destination node t either selects a route to s from its own cache or uses the route recorded in a request packet to send a **route-reply message**

Our proposal: route selection

- the source node s_i gathers **a set of alternative paths** on which to route traffic to t_i
- a *multicommodity flow algorithm* [Bie02, You95]: iteratively define a **metric** w over the edges and select at each iteration the **shortest path** from the source node s to the target node t
- **goal**: to **balance** the total accumulated flow on the edges of the network for a given graph $G = (V, E)$
- **parameters**: a weighing constant ϵ and the number of computation rounds $\mathcal{I}$

THE PATH-SELECTION ALGORITHM

1. $w_e := 1$ for each $\{v, w\} \in E$
2. For (s_c, t_c), set $x_e^c := 0$
3. For $\mathcal{I}$ iterations, do:
   - For each (s_c, t_c), compute the shortest path $p(s_c, t_c)$ w.r.t. w
   - Let y^c be the flow vector resulting from routing f_c units of flow on $p(s_c, t_c)$
   - For each $e \in E$, $x_e^c := x_e^c + y_e^c$,

$$w_e := \left(1 + \epsilon \sum_c y_e^c\right) w_e$$

4. $\boldsymbol{x} := \frac{1}{\mathcal{I}} \cdot \boldsymbol{x}$

IMPLEMENTATIONAL ISSUES

The flows y needed by the algorithm can be **embedded on standard** DSR **routing control packets**, allowing the intermediate nodes to update their w values **locally** as the route reply packets travel from the target node back to the source node.

$\mathcal{I}$ and ϵ are known by s_i and are be embedded on route-request packets.

ROUTING ON THE SET OF PATHS

- at the kth iteration, the algorithm selects a path p_i^k, which is *not necessarily distinct* from the paths selected earlier
- $\Rightarrow$ at iteration k, the source node is aware of at least one and at most k distinct paths to the destination
- s_i selects **uniformly at random** one of the k paths stored whenever it wishes to route a packet to t_i
- $\Rightarrow$ *multiple entries* in the table of $k \Rightarrow$ path "weights"
- routes are *never* deleted (in standard DSR, a collision triggers the removal of the from the cache)

NODE DUTIES: SOURCE NODE

s_i issues a request for iteration $k+1$ when it has received the replies for iteration k *or* after a timeout occurs.

A timeout is needed as requests may be lost.

In the worst case no reply will be received.

NODE DUTIES: INTERMEDIATE NODES

- need to store information on the balance request and reply messages
- requests and replies of different iterations for the same pair (s_i, t_i) may be circulating simultaneously
- when receiving a route request for s_i on iteration k, after having already forwarded one for that same iteration, must examine **the accumulated cost of the path** stored in the newly arrived request
- the new request is **forwarded only if it offers a better path** with respect to the metric w than the previously forwarded packets for the same iteration k

NODE DUTIES: TARGET NODE

- hop-counts of the paths on which the requests traveled do not necessarily match the order in which the requests arrive
- t_i initiates a **waiting period** in which it collects more balance requests for the s_i and iteration k when the first such packet arrives
- afterwards it sends a **route-reply message** of iteration k to s_i

EXPERIMENTS

- implemented the modified DSR in the ns-2 simulator [MFFV]
- compared the behavior of our approach to the performance of standard DSR
- Network: a 10×10-node grid with two source-destination pairs crossing
- **first**: **initialization phase** for the route selections
- **second**: **data transmission** for a period of 1,000 seconds
- **recorded**: the total number of packets received at the destinations per each time step

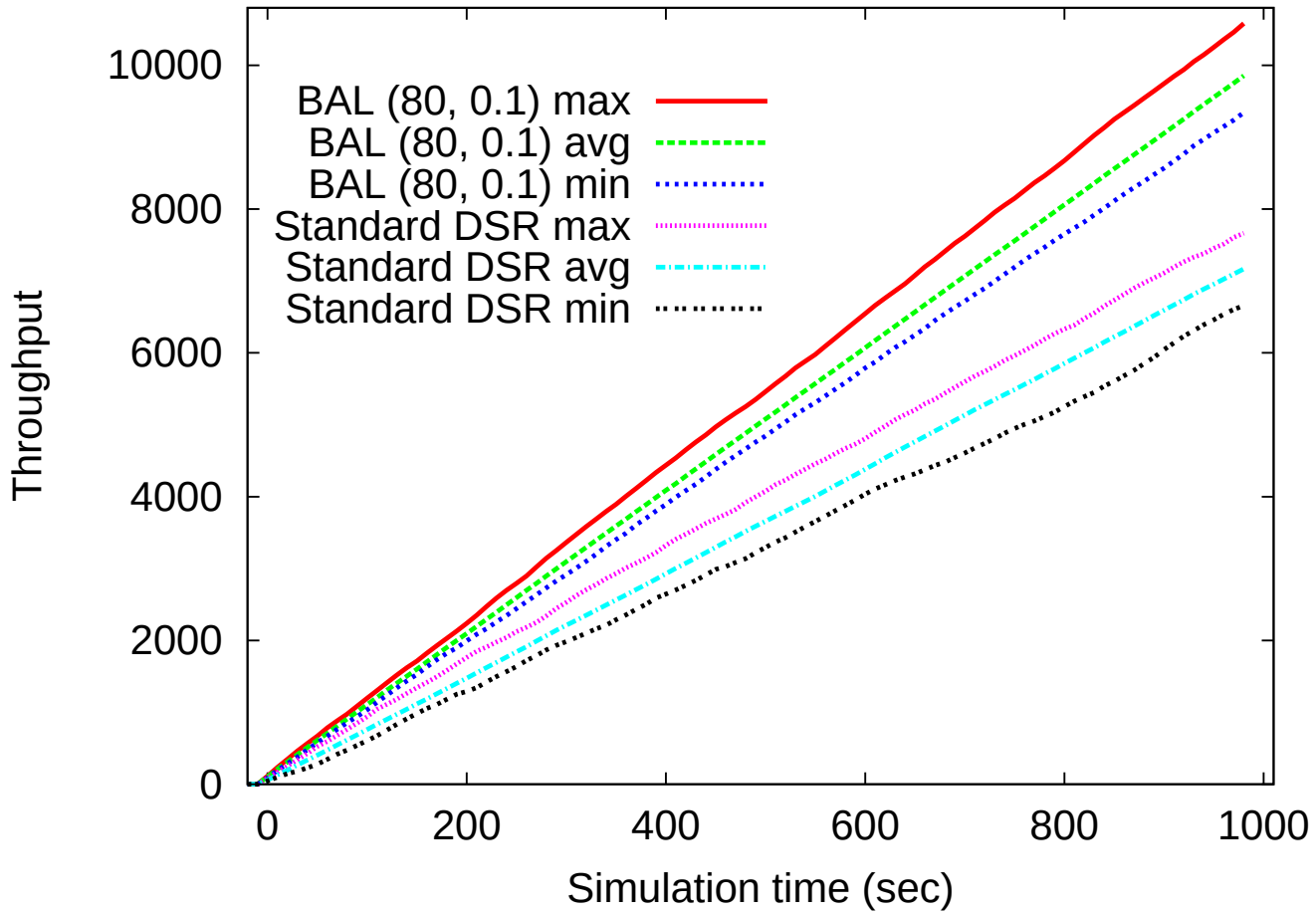
PARAMETER SELECTIONS

$$\begin{aligned} \mathcal{I} &\in \{5, 10, 20, 40, 80, 120, 160\}, \\ \epsilon &\in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.8\} \end{aligned}$$

10 repetitions for each setup (more are running as we speak)

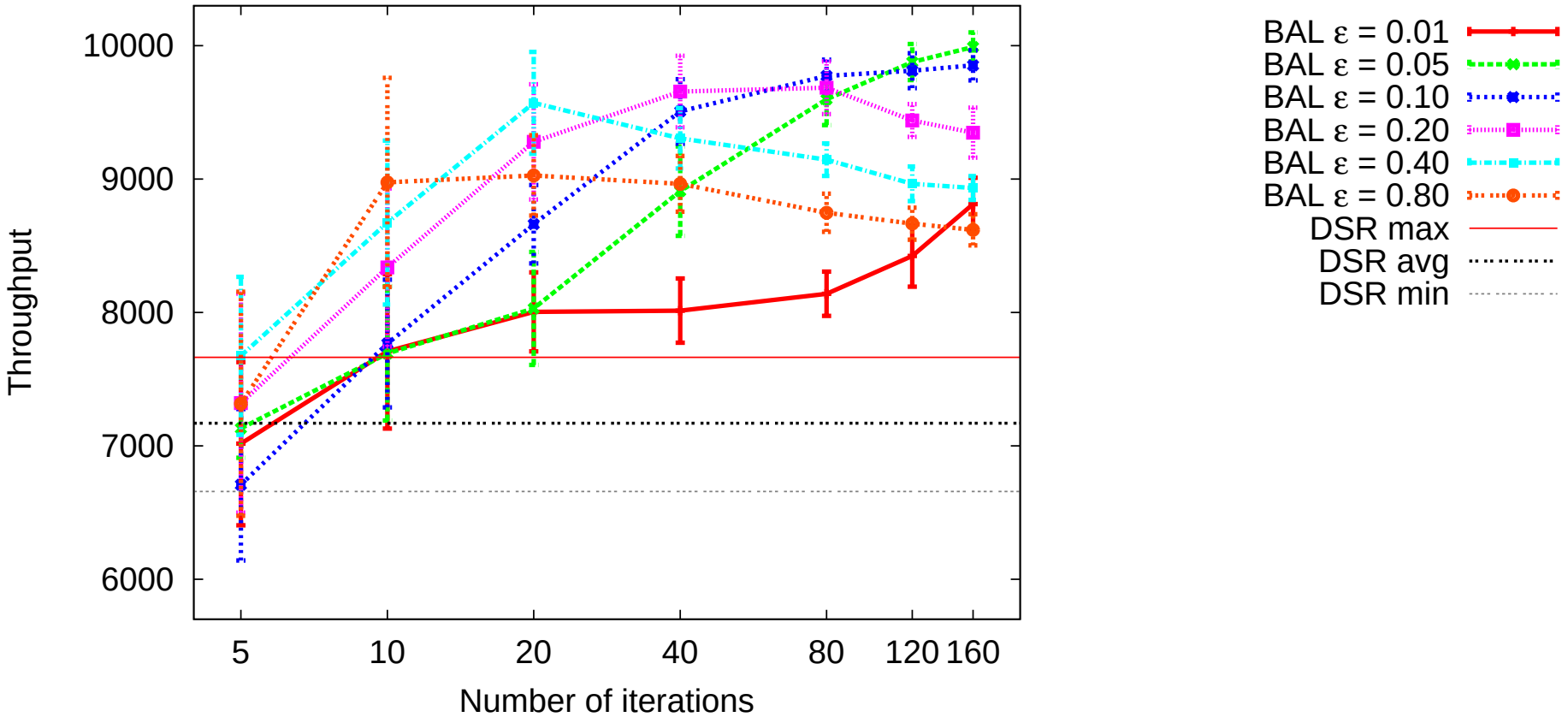
throughput $=$ the number of data packets multiplied by 2048

AN EXAMPLE $(\mathcal{I}, \epsilon)$ PAIR



Over 26 repetitions.

Throughput achieved at time $\mathcal{T} = 1{,}000$



CONCLUSIONS

- little incentive to use more than 40 iterations
- for smaller values of ϵ, high iteration counts yield greater improvements
- $\epsilon = 0.01$ seems to be so small that the algorithm would require a higher number of iterations to work well
- improvement in the throughput DSR can reach 35 percent (for $\epsilon = 0.05$ and $\mathcal{I} = 160$
- will be necessary to consider also the amount of effort needed in computing the iterations

Further work

- “fairness” of the routing
- elimination of the initial phase of the path-finding iterations altogether
- interleaving of the discovery of alternate routes with the data transmission along the original route
- different scenarios for network topologies, traffic patterns, and source-destination selections

COMMENTS, QUESTIONS?

This research was supported by the Academy of Finland, grant number 206235 (ANNE).

The ns-2 simulations and experiments are by André Schumacher.

Part of ongoing research with Pekka Orponen and Harri Haanpää.

References

[BBR04] Yair Bartal, John W. Byers, and Danny Raz. Fast, distributed approximation algorithms for positive linear programming with applications to flow control. SIAM *Journal on Computing*, 33(6):1261–1279, June 2004.

[Bie02] Daniel Bienstock. *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*, volume 53 of *International series in operations research & management science*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[CCK04] Han Choe, Seongho Cho, and Chong-kwon Kim. A dynamic mechanism for distributed optimization of

overlay multicast tree. In *International Conference on Information Networking (*ICOIN*) 2004*, February 2004.

[GK05] Craig Gotsman and Yehuda Koren. Distributed graph layout for sensor networks. *Journal of Graph Algorithms and Applications*, 9(3):327–346, 2005.

[GLBML01] Philippe Golle, Kevin Leyton-Brown, Ilya Mironov, and Mark Lillibridge. Incentives for sharing in peer-to-peer networks. In L. Fiege, G. Mühl, and U. Wilhelm, editors, *Electronic Commerce: Proceedings of the Second International Workshop (*WELCOM *2001)*, volume 2232 of *Lecture Notes in Computer Science*, pages 75–87, Heidelberg, Germany, November 2001. Springer-Verlag GmbH.

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch.

DSR: *The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison Wesley, Reading, MA, USA, 2001.

[JMH04] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks DSR, July 2004. Internet draft, `draft-ietf-manet-dsr-10.txt`.

[MFFV] Steven McCanne, Sally Floyd, Kevin Fall, and Kannan Varadhan. The network simulator `ns-2`. The VINT project, available for download at `http://www.isi.edu/nsnam/ns/`.

[MLL05] Ritesh Madan, Zhi-Quan Luo, and Sanjay Lall. A distributed algorithm with linear convergence for maximum lifetime routing in wireless sensor networks.

In *Proceedings of the Allerton Conference on Communication, Control and Computing*, September 2005.

[MS05] Evangelos Markakis and Amin Saberi. On the core of the multicommodity flow game. *Decision Support Systems*, 39(1):3–10, March 2005.

[SP03] Jeffrey Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In *Proceedings of the Second International Workshop on Peer-to-Peer Systems (*IPTPS*'03)*, volume 2735 of *Lecture Notes in Computer Science*, pages 139–148, Heidelberg, Germany, 2003. Springer-Verlag GmbH.

[You95] Neal E. Young. Randomized rounding without solving the linear program. In *Proceedings of the Sixth Annual*

ACM-SIAM *Symposium on Discrete Algorithms*, pages 170–178. ACM/SIAM, 1995.