A Study of Shared-Memory Mutual Exclusion Protocols using CADP

Radu Mateescu and Wendelin Serwe

INRIA Rhône-Alpes / VASY

http://www.inrialpes.fr/vasy

Overview

- Mutual exclusion on shared-memory machines
- Formal specification using LOTOS NT
- Functional analysis by model checking using MCL
- Performance evaluation using IMCs
- Conclusion and future work

Mutual exclusion on shared-memory machines

- Long-standing problem in concurrent programming [Dijkstra-65]:

  Protect a shared resource against concurrent non-atomic accesses from competing processes, which communicate by atomic read/write operations on shared variables

- Mutual exclusion protocols:
  - Ensure that at most one process accesses the resource
  - Guarantee the progress of execution
- Dozens of protocols proposed in the literature (see survey in [Anderson-Kim-Herman-03])
- Performance assessment mainly by experimental measures

  ➔ *our goal: provide model-based quantitative analysis*

Mutual exclusion protocols

- Structure of a process P competing for the access to the shared resource:

```
loop
  non critical section ;     → may loop forever
  entry section ;            → access shared variables
  critical section ;         → access resource
                             → must terminate
  exit section               → access shared variables
end loop
```

Study of 24 protocols (for two processes)

- black-white bakery protocol [Taubenfeld-04]
- Burns & Lynch [80]
- Craig and Landin & Hagersten (CLH) [93,94]
- Dekker [68]
- Dijkstra [65]
- Kessels [82]
- Knuth [66]
- Lamport [87]
- Mellor-Crummey & Scott [91]
- Peterson [81]
- Szymanski [88]
- 12 automatically generated protocols [Bar-David-Taubenfeld-03]
- trivial (incorrect) one-bit protocol (for benchmarking)

Interactive Markov Chains
[Hermanns-02]

- Single model for both
  - functional verification:
    extension of labeled transition systems (hide delay)
  - performance analysis:
    extension of Markov chains (hide actions)
- Enrich functional model with (exponential) delays by composition with additional processes
- Tool support by CADP (*http://vasy.inria.fr/cadp*)
  - functional verification:
    generator, evaluator, bisimulator, bcg_min, ...
  - performance analysis:
    bcg_steady, bcg_transient, bcg_min, ...

LOTOS NT
[Champelovier-Clerc-Garavel-et-al-10]

- Integration of the features of
  - process algebras
  - imperative programming languages
- User-friendly syntax and formal semantics
- Input language of CADP (Int.open)
  - compilation into LOTOS
  - generation of the labeled transition system (LTS)
  - connection to on-the-fly exploration tools

Knuth's protocol [Knuth-66]

three shared variables A[0], A[1], B

Process P_i

```
loop
        non critical section ;
        loop
           A[i] := 1 ;
           await B == i or A[j] == 0 ;
           A[i] := 2 ;
           if A[j] != 2 then break ;
        end loop ;
        B := i ;
        critical section ;
        B := j ;
        A[i] := 0
end loop
```

$i \in \{ 0, 1 \}$
other process: $j = 1-i$

entry section

exit section

Knuth's protocol in LOTOS NT

```
process P [NCS:Pid, CS:Access, A, B:Operation] (i:Nat) is
  var k, a_k, b: Nat in k := 1 - i;
    loop
      NCS (i);
      loop L1 in
        A (Write, i, 1 of Nat, i);
        loop L2 in
          B (Read, ?b, i); A (Read, j, ?a_j, i);
          if (b == i) or (a_j == 0) then break L2 end if
        end loop;
        A (Write, i, 2 of Nat, i);
        A (Read, j, ?a_j, i); if a_j != 2 then break L1 end if
      end loop;
      B (Write, i, i);                                  entry section
      CS (Enter, i); CS (Leave, i);
      B (Write, j, i);
      A (Write, i, 0 of Nat, i)                         exit section
    end loop
end var end process
```

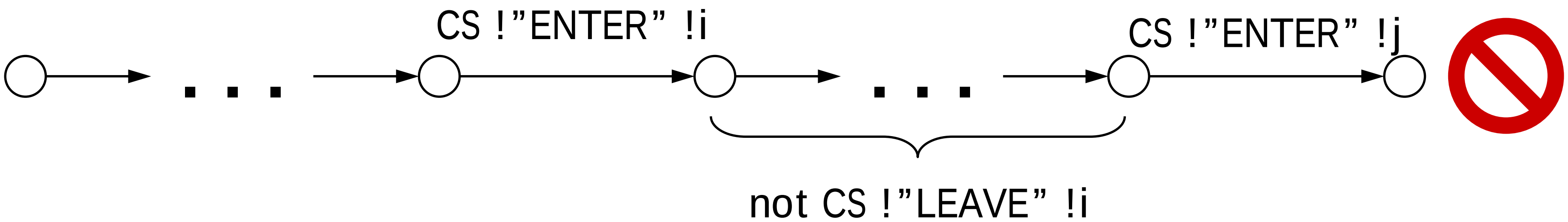
MCL (Model Checking Language)
[Mateescu-Thivolle-08]

- Extension of modal μ-calculus with:
  - Regular expressions over action sequences [Mateescu-Sighireanu-03]
  - Modalities that extract data values from LTS labels
  - Fixed point operators parameterized by data variables
  - Constructs inspired from programming languages
- Tool support: EVALUATOR 4.0
  - On-the-fly verification of MCL formulas on LTSs
  - Diagnostic generation (examples and counterexamples)
  - Reusable libraries of derived operators (CTL, ACTL, …) and property patterns [Dwyer-et-al-99]

Mutual exclusion *(safety)*

Two processes can never execute simultaneously their critical sections.

```
[ true* .
  { CS !"ENTER" ?i:Nat } .
  (not { CS !"LEAVE" !i })* .
  { CS !"ENTER" ?j:Nat where j <> i }
] false
```



Livelock freedom *(liveness)*
(first formulation)

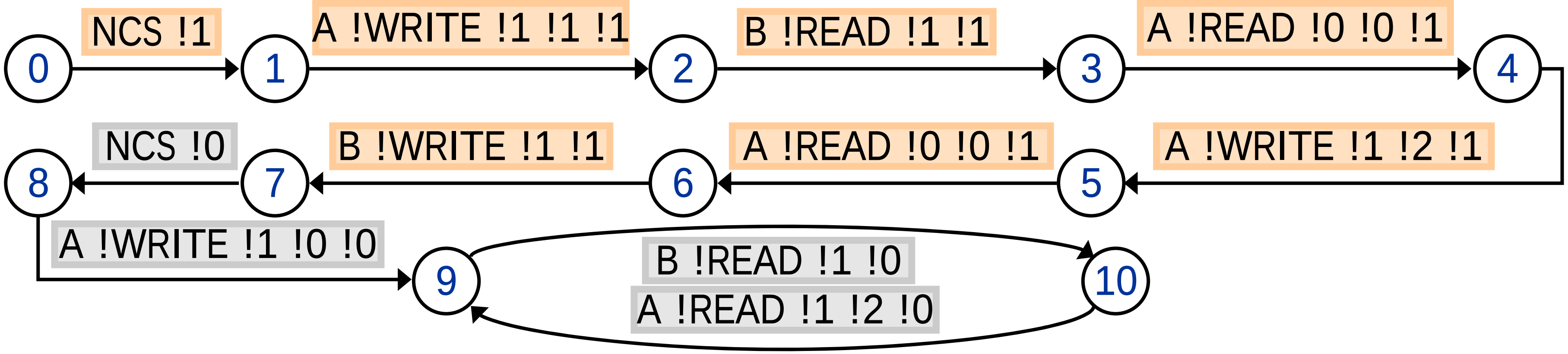
Each time a process is in its entry section, then some process will eventually enter its critical section.

```
[  true* . { NCS ?i:Nat } .
   (not { ?any ?"READ"|"WRITE" ... !i })* .
   { ?any ?"READ"|"WRITE" ... !i }
]  mu X . (< true > true and
             [ not { CS !"ENTER" ?any } ] X)
```

➔ *this formula fails on all mutex protocols!*

Livelock freedom (first formulation)

Counterexample for Knuth's protocol:




```
loop
  non critical section ;
  loop
    A[0] := 1 ;
    await B == 0 or A[1] == 0 ;
    A[0] := 2 ;
    if A[1] != 2 then break ;
  end loop ;
  B := 0 ;
  critical section ;
  B := 1 ; A[0] := 0
end loop
```

P_0

```
loop
  non critical section ;
  loop
    A[1] := 1 ;
    await B == 1 or A[0] == 0 ;
    A[1] := 2 ;
    if A[0] != 2 then break ;
  end loop ;
  B := 1 ;
  critical section ;
  B := 0 ; A[1] := 0
end loop
```

P_1

Livelock freedom (second formulation)

Starvation freedom *(fairness)*

There is no cycle in which every process executes an instruction but no one enters its critical section.

```
[  true* . { NCS ?i:Nat } .
   (not { ?any ?"READ"|"WRITE" ... !i })* .
   { ?any ?"READ"|"WRITE" ... !i }
] not < (not { CS ... !i })* .
        { ?G:String ... ?j:Nat where ((G <> "CS") or (j <> i))} .
        (not { CS ... !i })* .
        { ?G:String ... !1-j where ((G <> "CS") or (j <> 1-i))}
      > @
```

➔ *holds on all mutex protocols*

➔ *holds on* some *mutex protocols*

Bounded overtaking

How many times a process can be overtook by the other one in accessing the critical section?

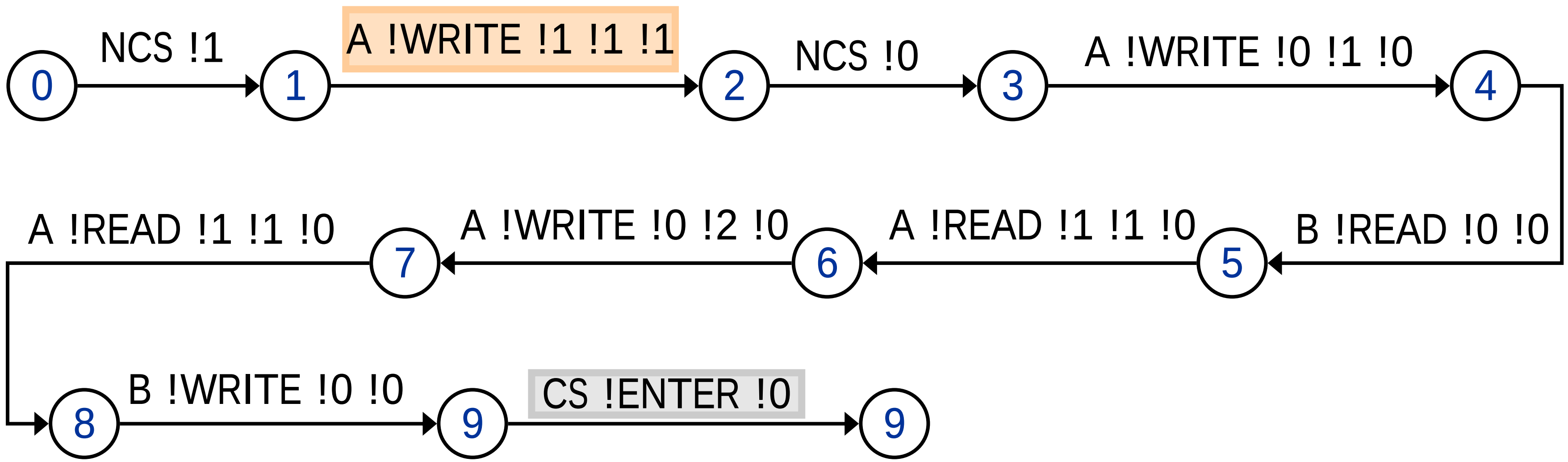
```
let max_overtaking:Nat := max in
   < true* . { NCS !1 } .
      (not { ?any ?"READ"|"WRITE" ... !1 })* .
      { ?any ?"READ"|"WRITE" ... !1 } .
      (   (not { CS ?any !1 })* .
          { ?G:String ... !1 where G <> "CS" } .
          (not { CS ?any !1 })* . { CS !"ENTER" !0 }
      ) { max_overtaking }
   > true
end let
```

P_0 overtakes P_1

Witness of bounded overtaking

For Knuth's protocol:

- witness with one overtake of P_1 by P_0 (max = 1):



- no witness with more overtakes of P_1 by P_0 found

Model checking summary

Protocol (2 processes)	Nb. vars.	IMC size states	IMC size trans.	L/S-free	Overtaking P_0/P_1	Overtaking P_1/P_0
1b_p1	1	89	130		1	1
burns_lynch	2	259	368	L	∞	3
szymanski		547	803		2	1
2b_p1		259	369	L	∞	1
2b_p2		271	386	L	∞	1
2b_p3		277	392	L	1	∞
dekker	3	599	856		4	4
knuth		917	1312		1	1
3b_p1		486	690		3	3
3b_p2		627	879	L	∞	1
peterson		407	580		2	2
3b_c_p1		627	884		2	2
3b_c_p2		407	580		2	2
3b_c_p3		363	516		2	2

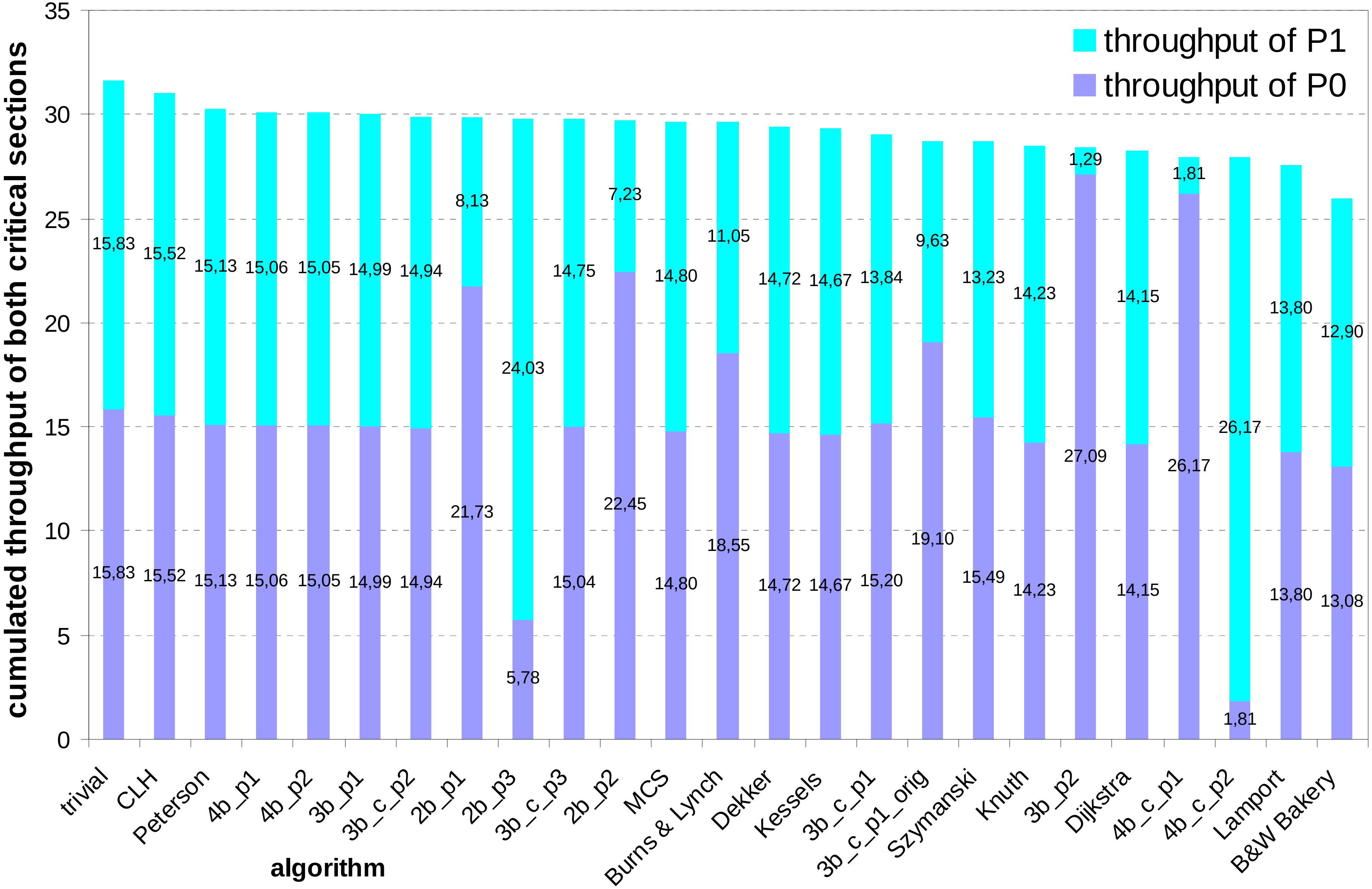
Protocol (2 processes)	Nb. vars.	IMC size states	IMC size trans.	L/S-free	Overtaking P_0/P_1	Overtaking P_1/P_0
lamport	4	1599	2274	L	∞	∞
kessels		1073	1502		2	2
clh		690	936		2	2
4b_p1		432	610	L	∞	1
4b_p2		871	1229		3	3
4b_c_p1		1106	1542	L	∞	1
4b_c_p2		1106	1542	L	1	∞
dijkstra	5	899	1260	L	∞	∞
mcs		424	612		2	2
bw_bakery	7	31222	43196		2	2

mutual exclusion and *livelock freedom*
satisfied by all protocols

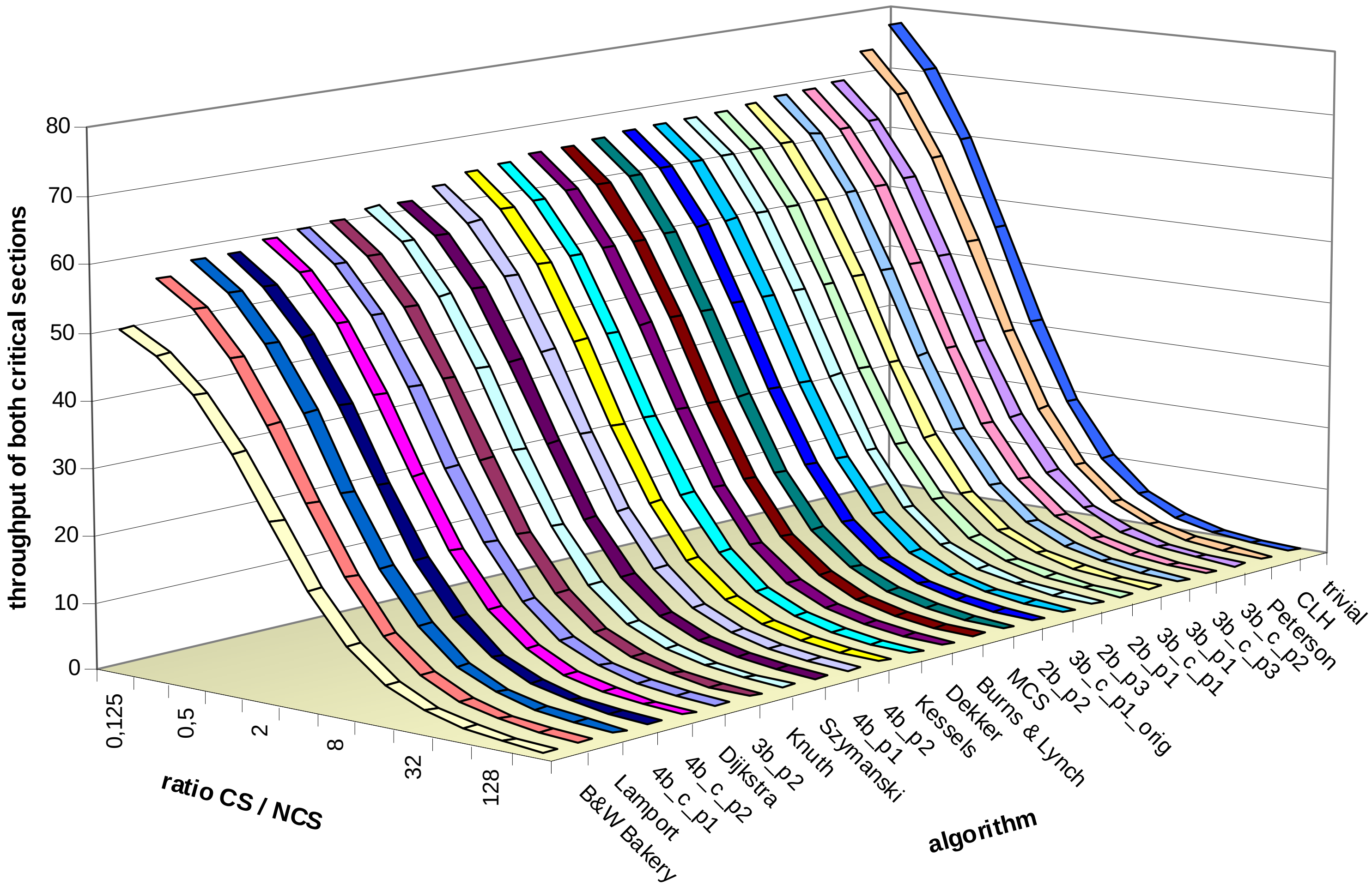
Performance experiments

- Goal: detect tendencies, no absolute values
- Throughput of the critical section:
  - relative (one process only)
  - cumulative (sum of both processes)
- Rate parameters common to all experiments:
  - read access: 3000
  - write access: 2000 (also for fetch&store, compare&swap)
  - critical section: 100
- Varying rate for the non-critical section(s)

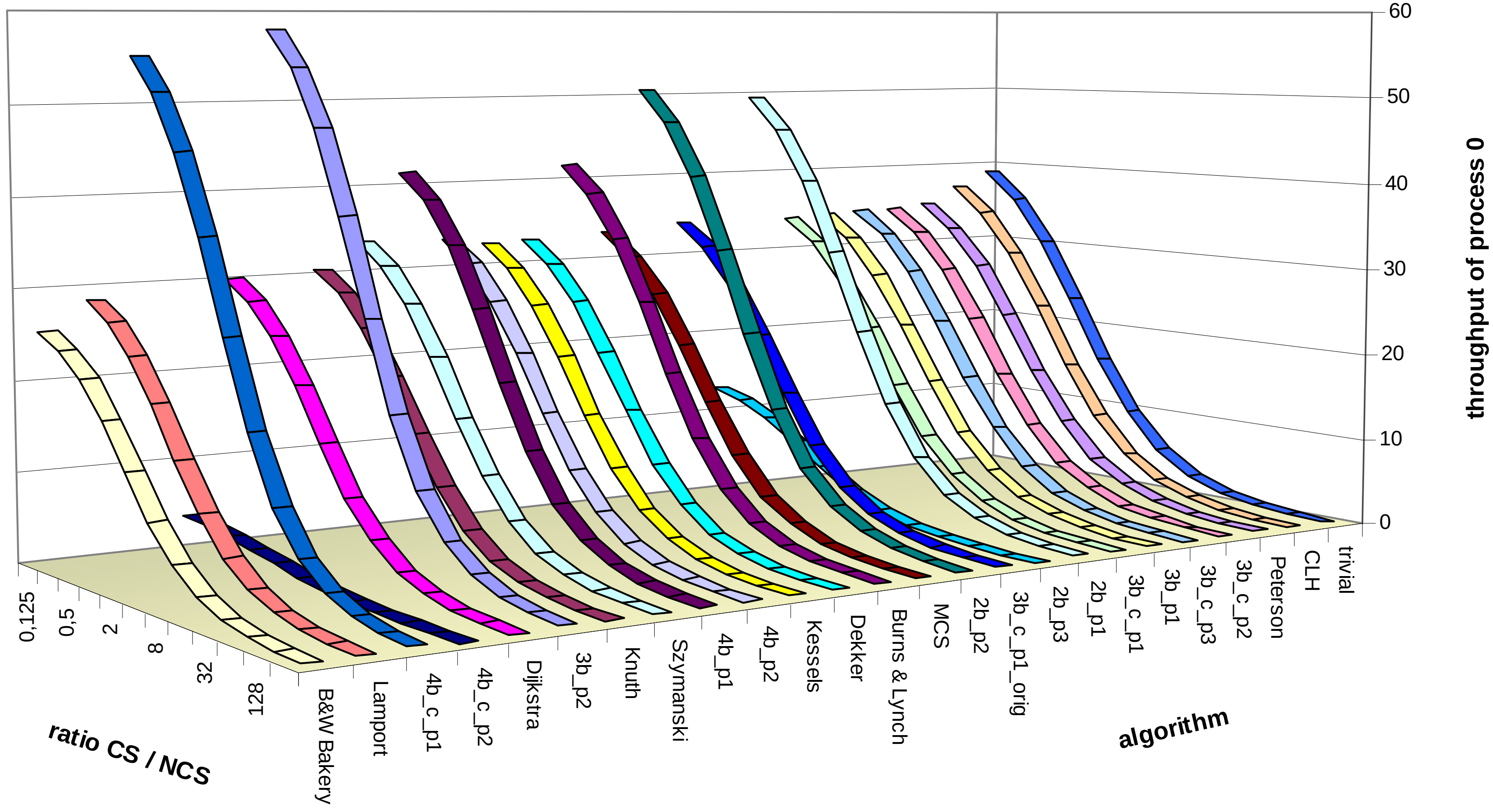
Comparison of the protocols



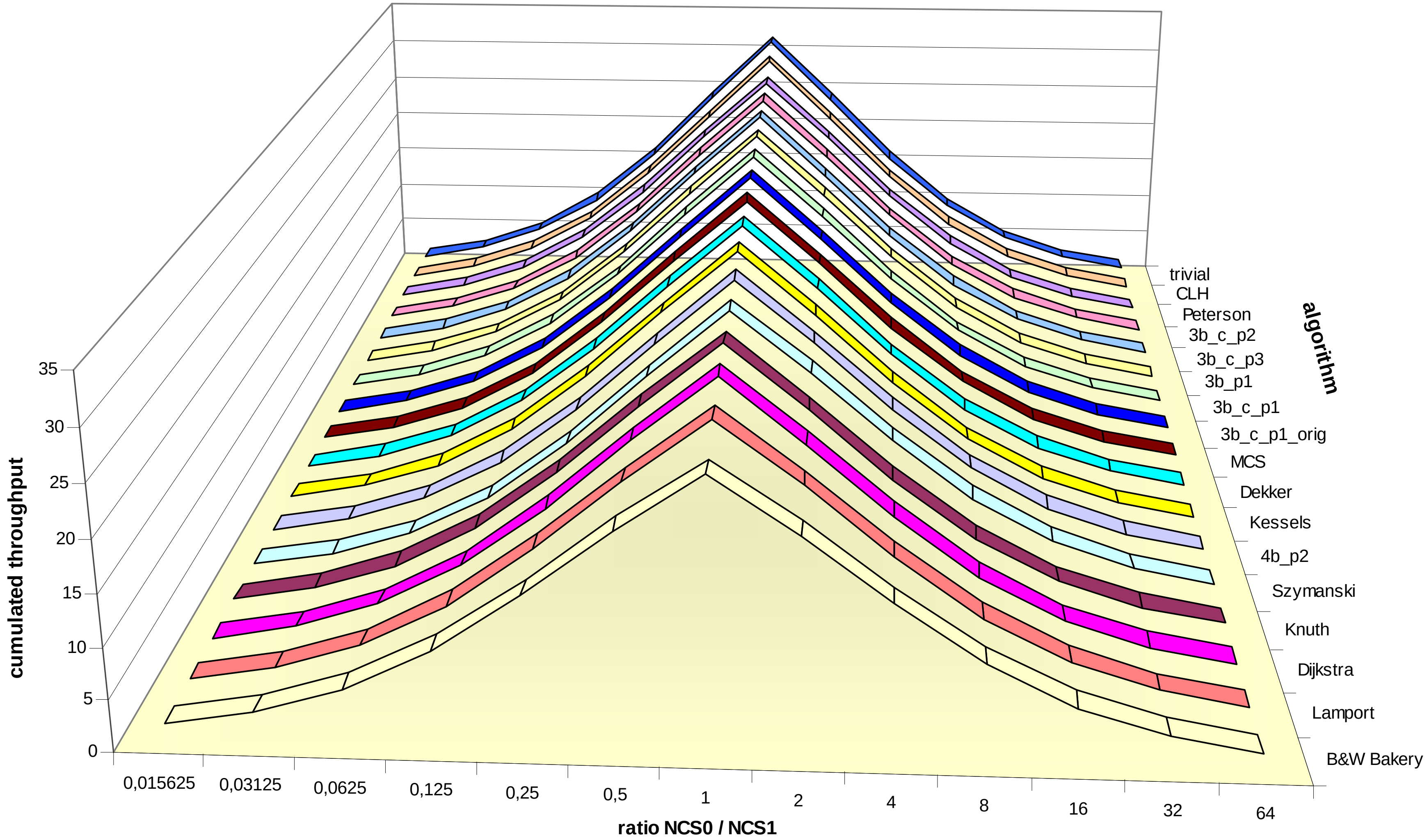
Varying ratio critical/non-critical section



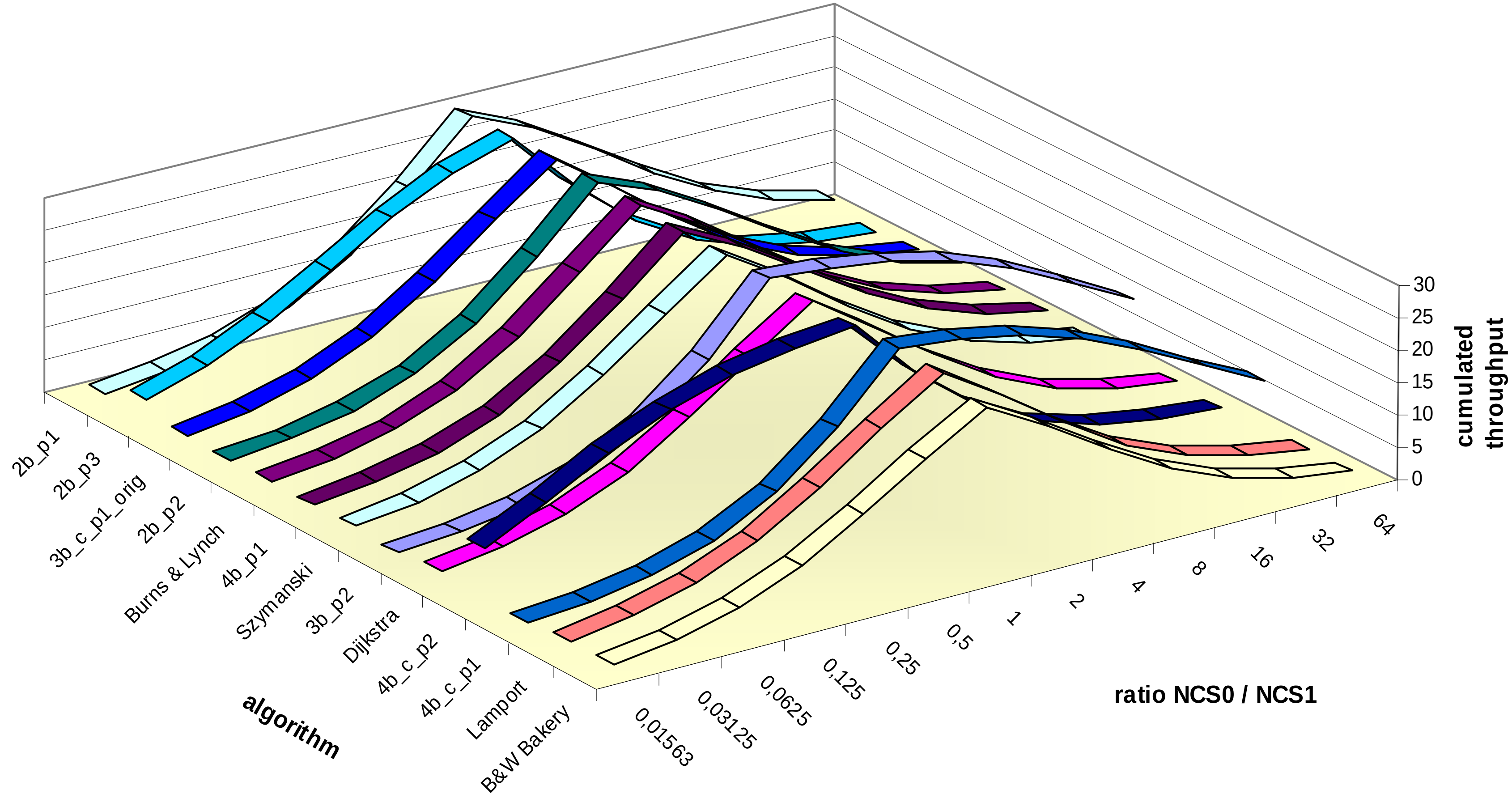
Varying ratio critical/non-critical section



Varying ratio of non-critical sections
(cumulated throughput for symmetric protocols)

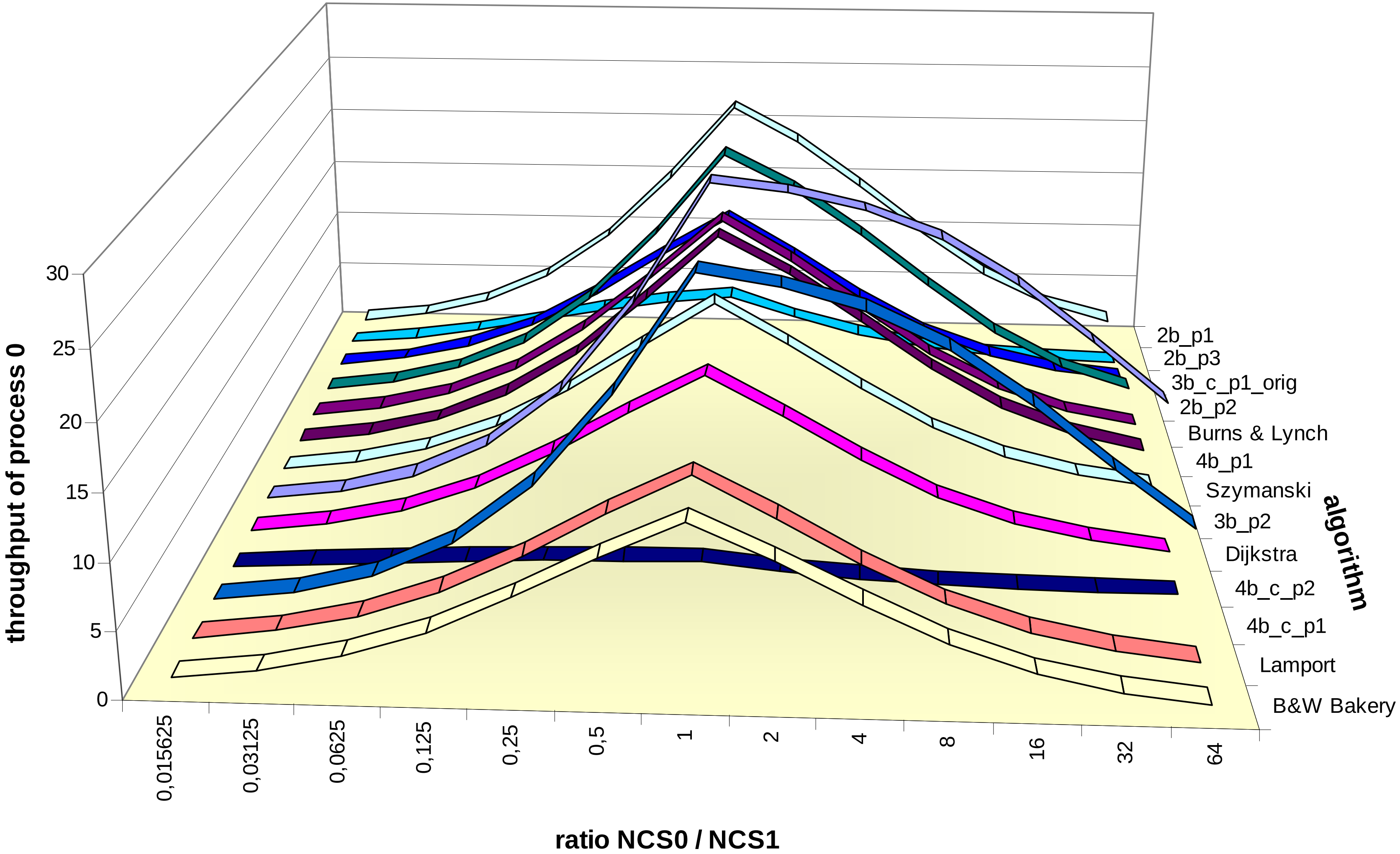


Varying ratio of non-critical sections
(cumulated throughput for asymmetric protocols)



Varying ratio of non-critical sections

(throughput of process 0 for asymmetric protocols)



Conclusion and future work

- Formal analysis and performance evaluation of mutual exclusion protocols on a single model
- Automated analysis using CADP tools

- Extend performance study to
  - More than two processes
  - Determine variable placement:

    ➔ *frequent accesses should be local, not remote*
  - Better handling of nondeterminism:

    based on new techniques proposed by Zhang&Neuhäußer: Model-checking Interactive Markov Chains, TACAS 2010

More information about CADP

http://vasy.inria.fr/cadp

and

http://cadp.forumotion.com