コンピュータ科学特別講義Ⅳ

Parallel Algorithm Design (#9)

Masato Edahiro

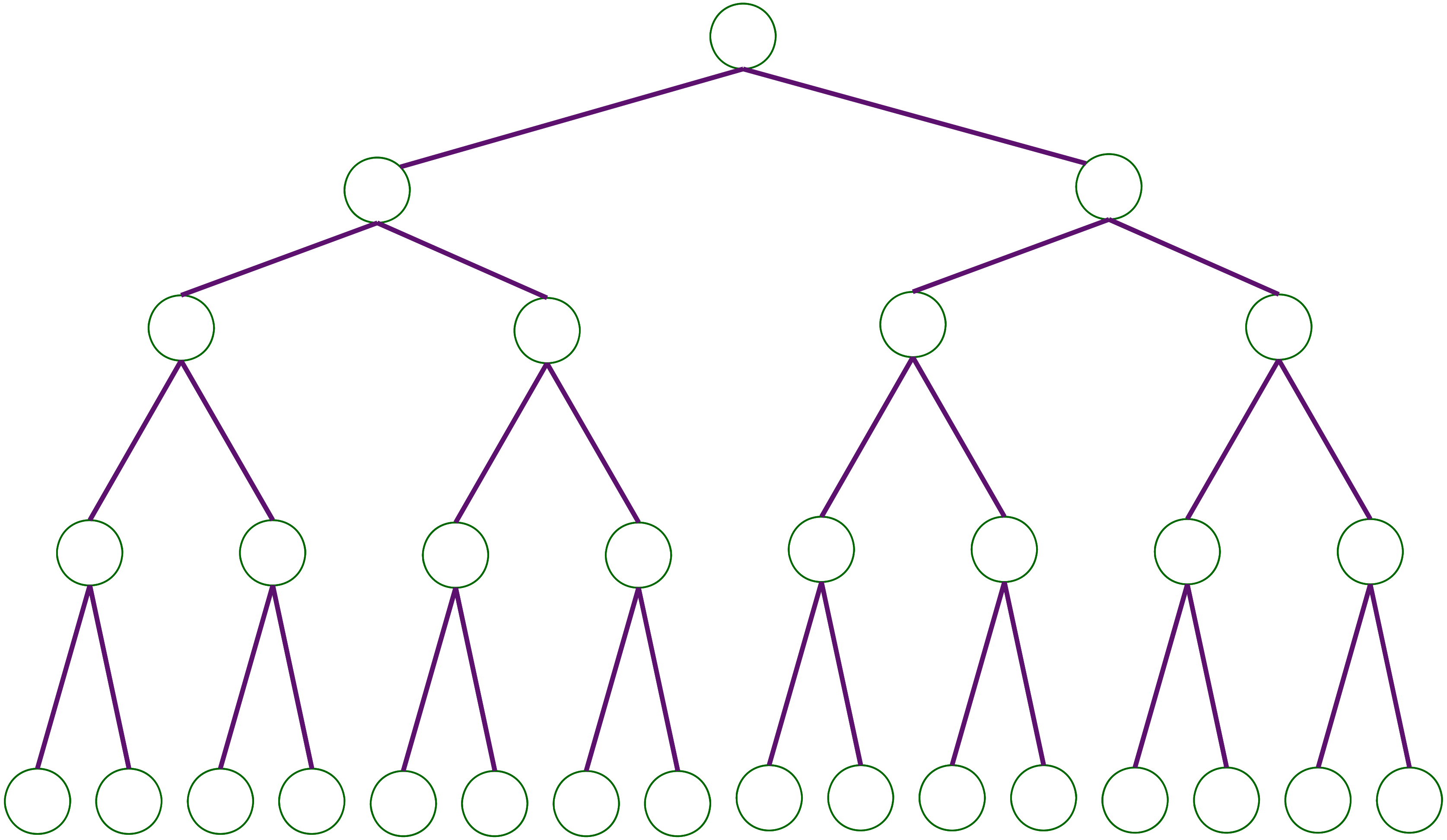
July 13, 2018

Please download handouts before class from
http://www.pdsl.jp/class/utyo2018/

Contents of This Class

- Our Target
  - Understand Systems and Algorithms on “Multi-Core” processors
- Schedule (Tentative)
  - #1 April 6 (=Today) What’s “Multi-Core”?
  - #2 April 13 : Parallel Programming Languages (Ex. 1)
  - April 20, 27, May 4, 11, 18: NO CLASS
  - #3 May 25 : Parallel Algorithm Design
  - #4 June 1 (Fri) : Laws on Multi-Core
  - #5 June 8 : Examples of Parallel Algorithms (1) (Ex. 2)
  - June 15: NO CLASS
  - #6 June 22 : Examples of Parallel Algorithms (2)
  - #7 June 29 : Examples of Parallel Algorithms (3)
  - #8 July 6 : Examples of Parallel Algorithms (4)
  - #9 July 13 : Examples of Parallel Algorithms (5) (Ex. 3)
  - (July 20)
  - If you want to graduate in August, ask Edahiro asap.

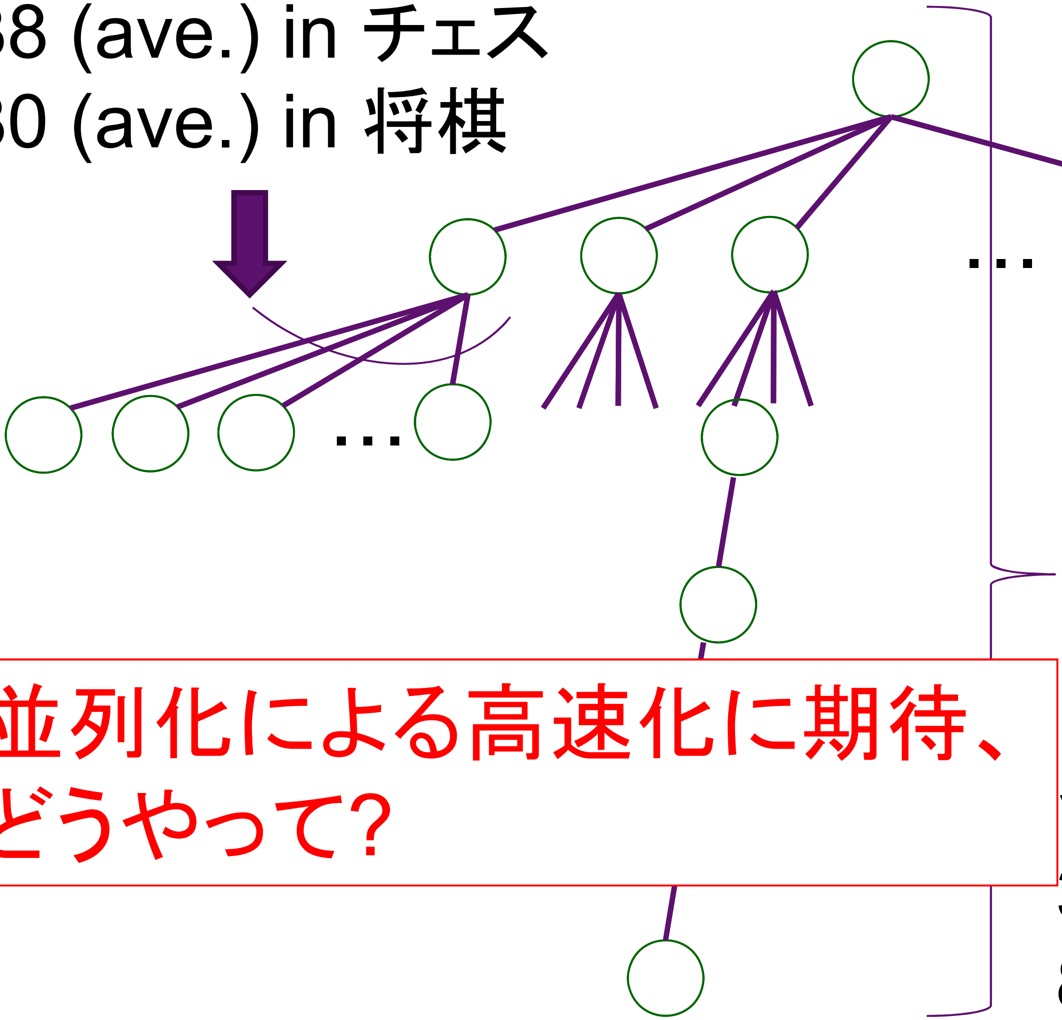
木構造探索



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

候補はどのくらいか

木の分岐数 (b)
38 (ave.) in チェス
80 (ave.) in 将棋



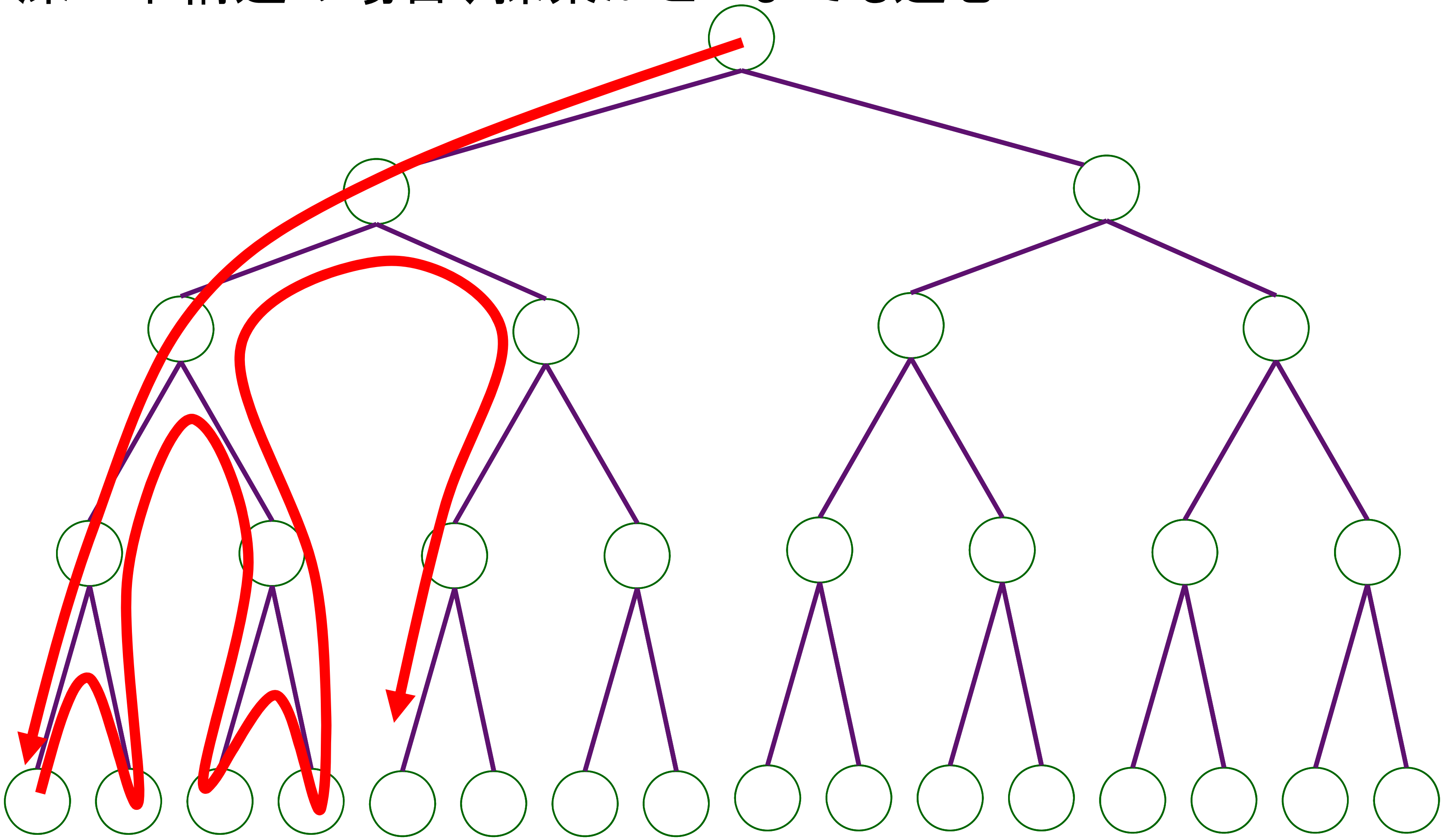
木の深さ (d)
80 (ave.) in チェス
115 (ave.) in 将棋

並列化による高速化に期待、
どうやって?

全体
38^{80} (ave.) in チェス
80^{115} (ave.) in 将棋

深さ優先探索

メモリ使用量は少なくてすむ
深い木構造の場合、探索がどこまでも進む

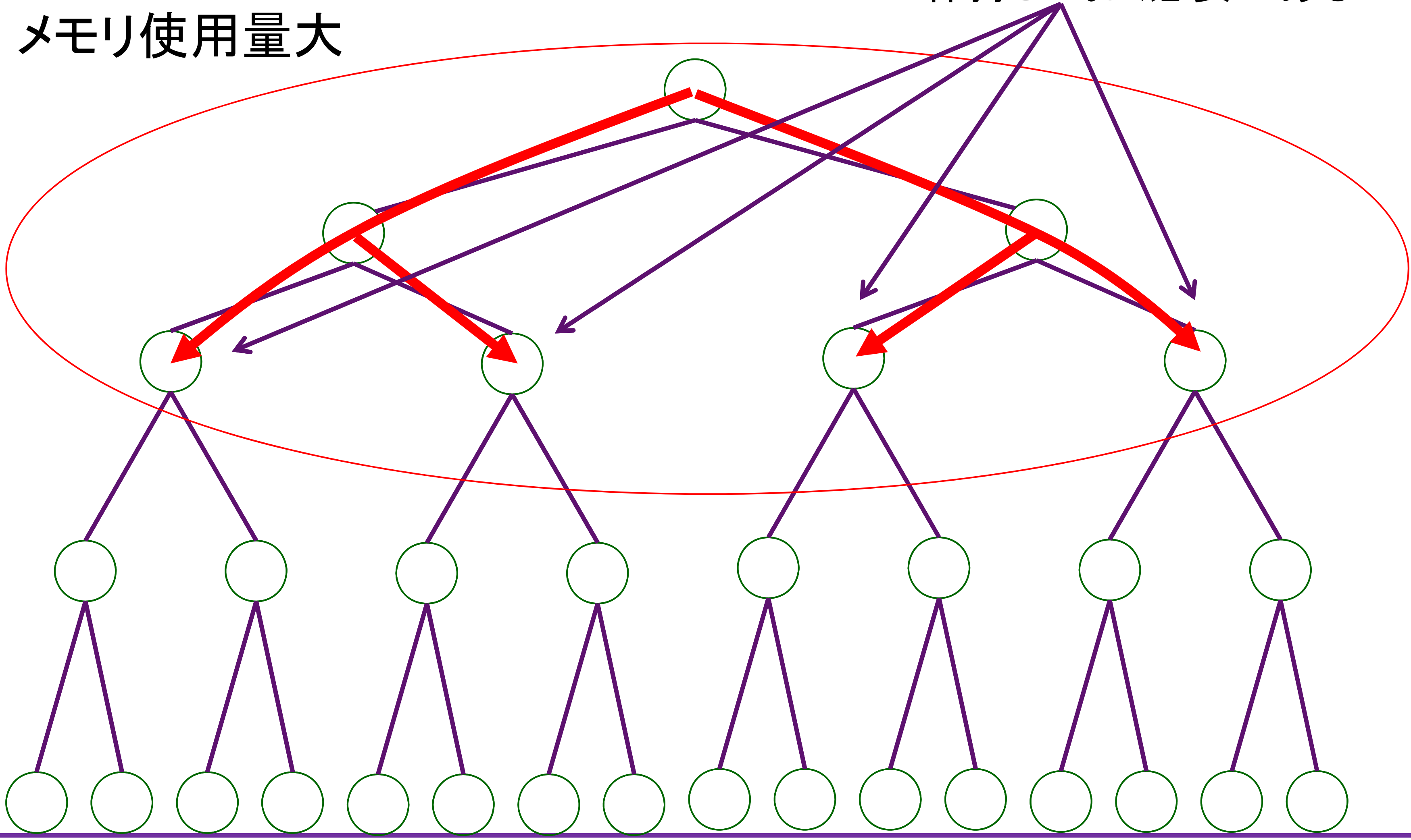


Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

幅優先探索

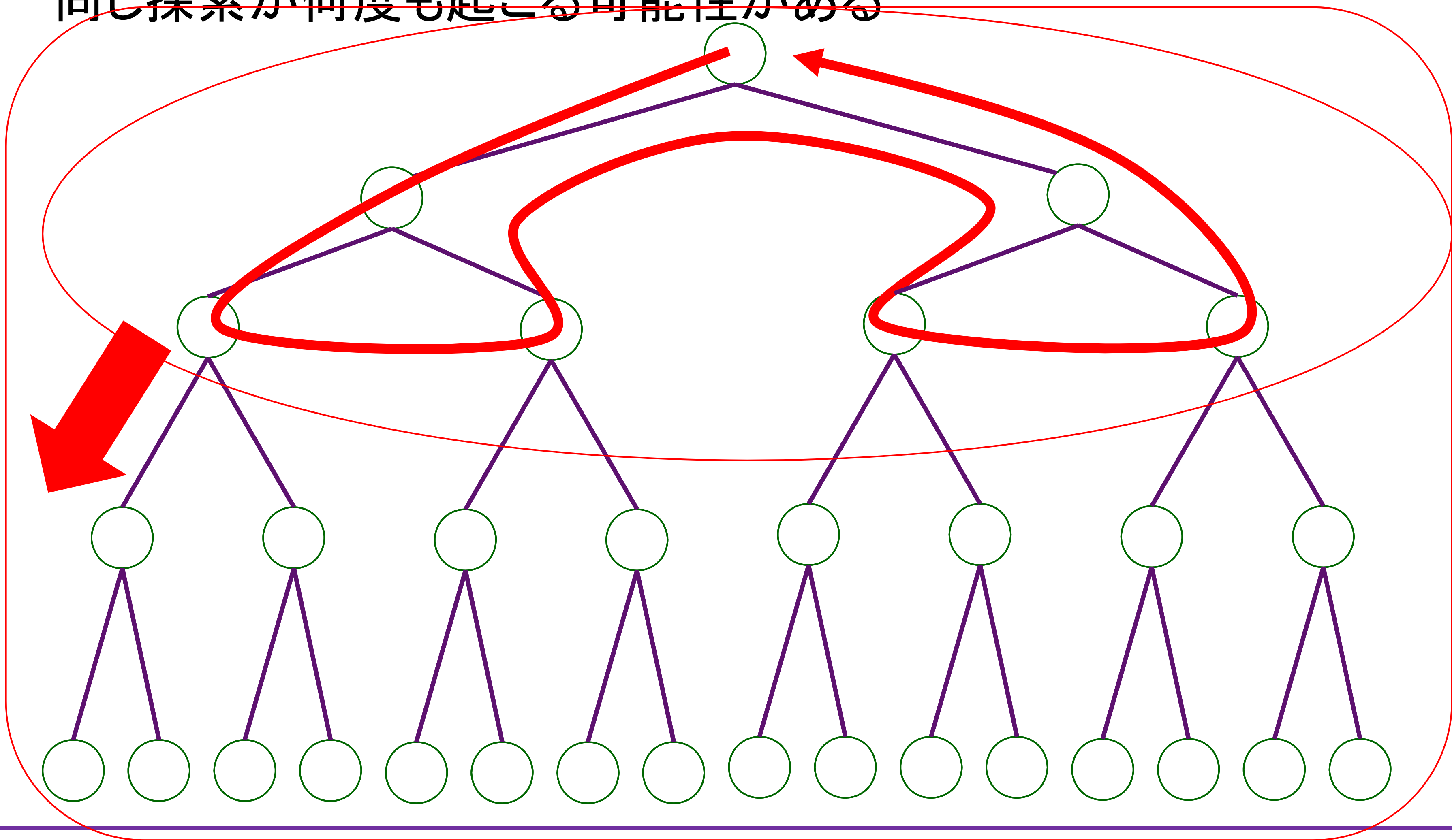
解より深くは探索しない
メモリ使用量大

すべての中間状態を
保持しておく必要がある



反復深化

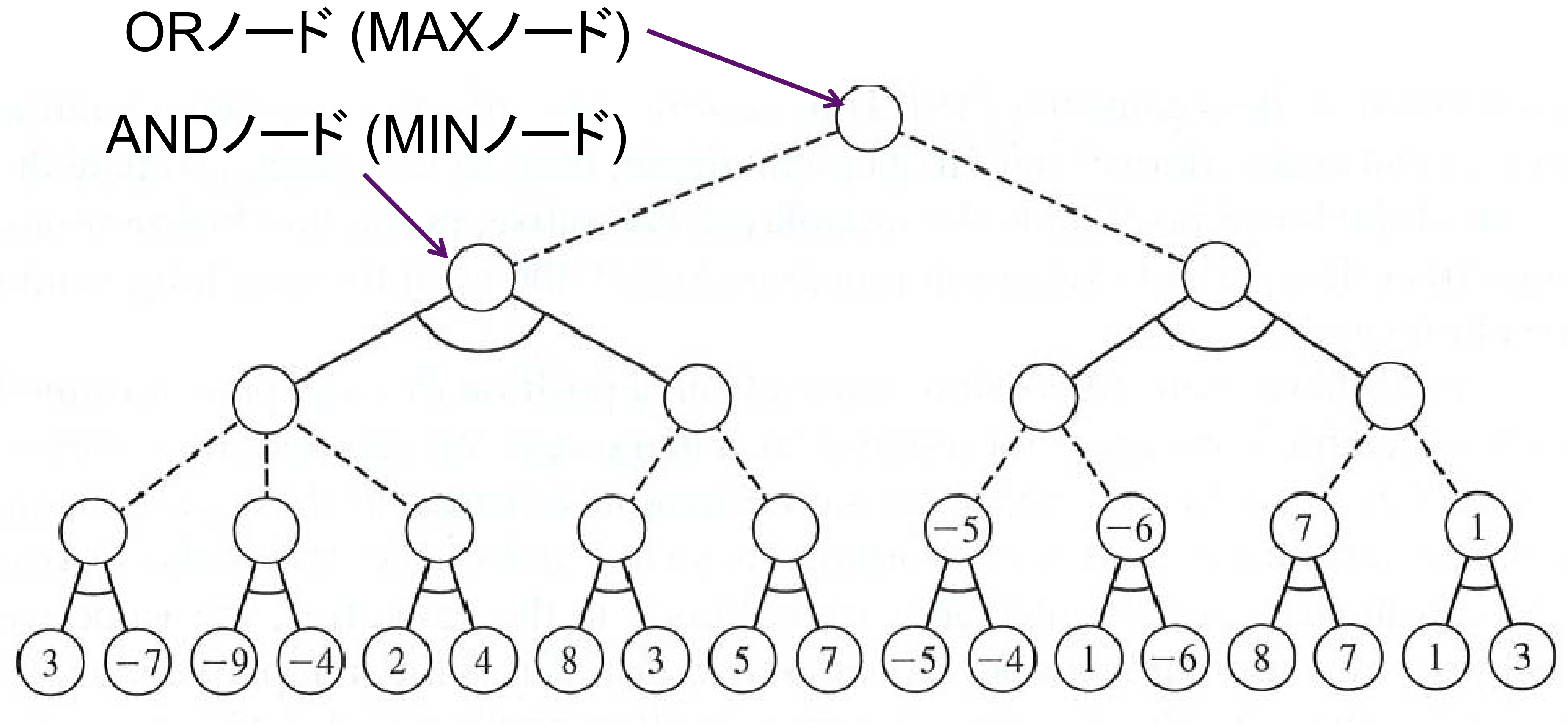
メモリ使用量小、解より深い探索が進み過ぎない
同じ探索が何度も起こる可能性がある



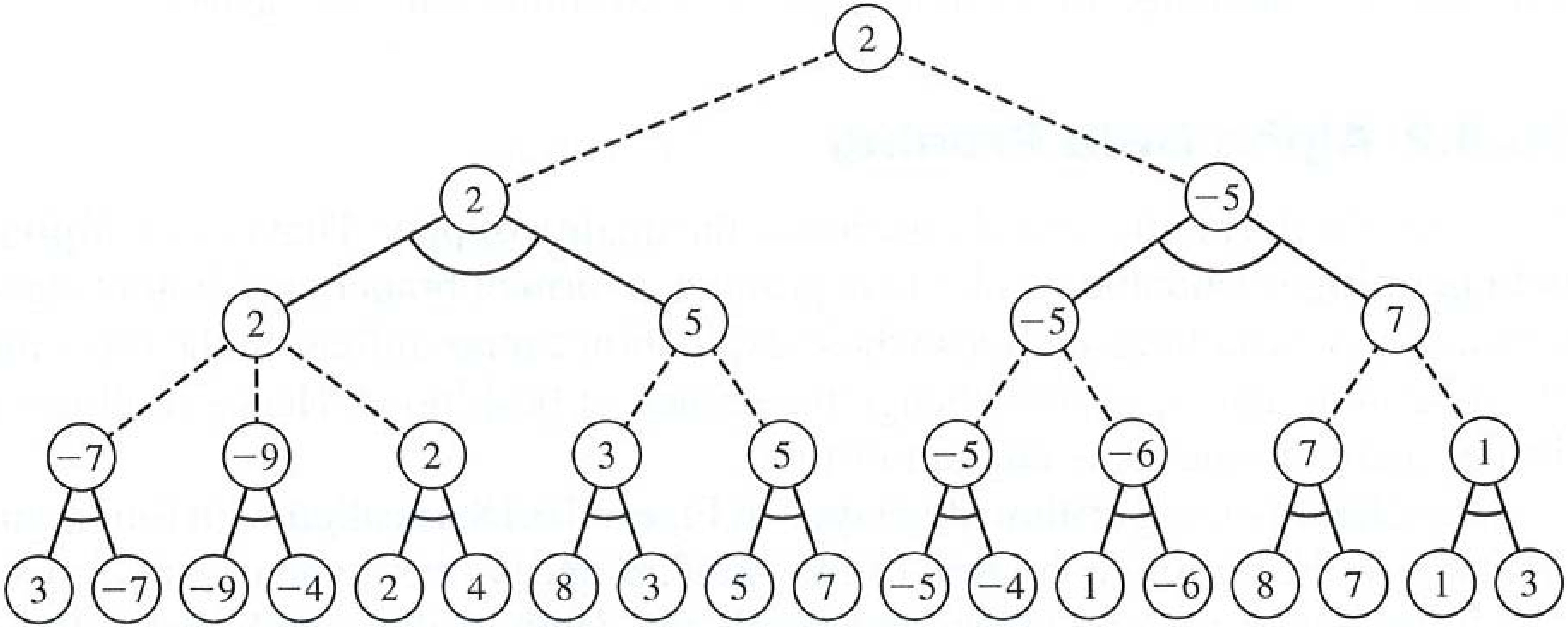
ゲーム木探索

- AND-OR木
  - ノードの値：（第一プレイヤの）ゲイン・利得
  - ORノード (MAXノード)
    - 第一プレイヤの順番
    - 第一プレイヤはゲインを最大化したい (MAX)
    - 第一プレイヤは最も良いノードを選択、すなわちすべての条件を考慮する必要はない (OR)→勝てるパスが一つでもあれば、そこに突き進めばよい
  - ANDノード (MINノード)
    - 第二プレイヤの順番
    - 第二プレイヤはゲインを最小化したい (MIN)
    - 第一プレイヤはすべての可能性を考慮する必要がある(AND)→負けるパスが一つでもあればそこに進まれるので、すべての可能性で負けないことを確認

ゲーム木の例



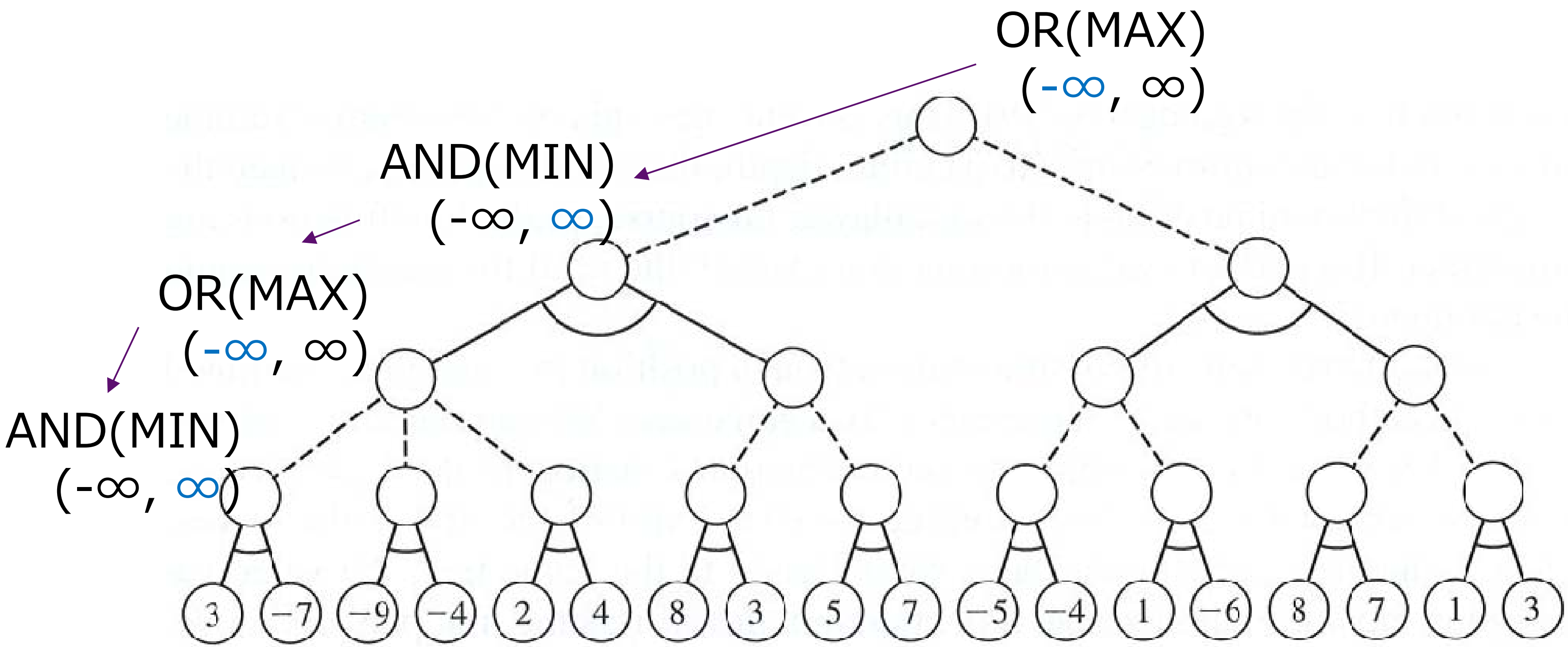
Minimaxアルゴリズム



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

Alpha-Beta枝刈り手法 (1)

- (α, β)
  - 探索を行うべき値（ゲイン）の範囲
  - 初期値 = (-∞, ∞)



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

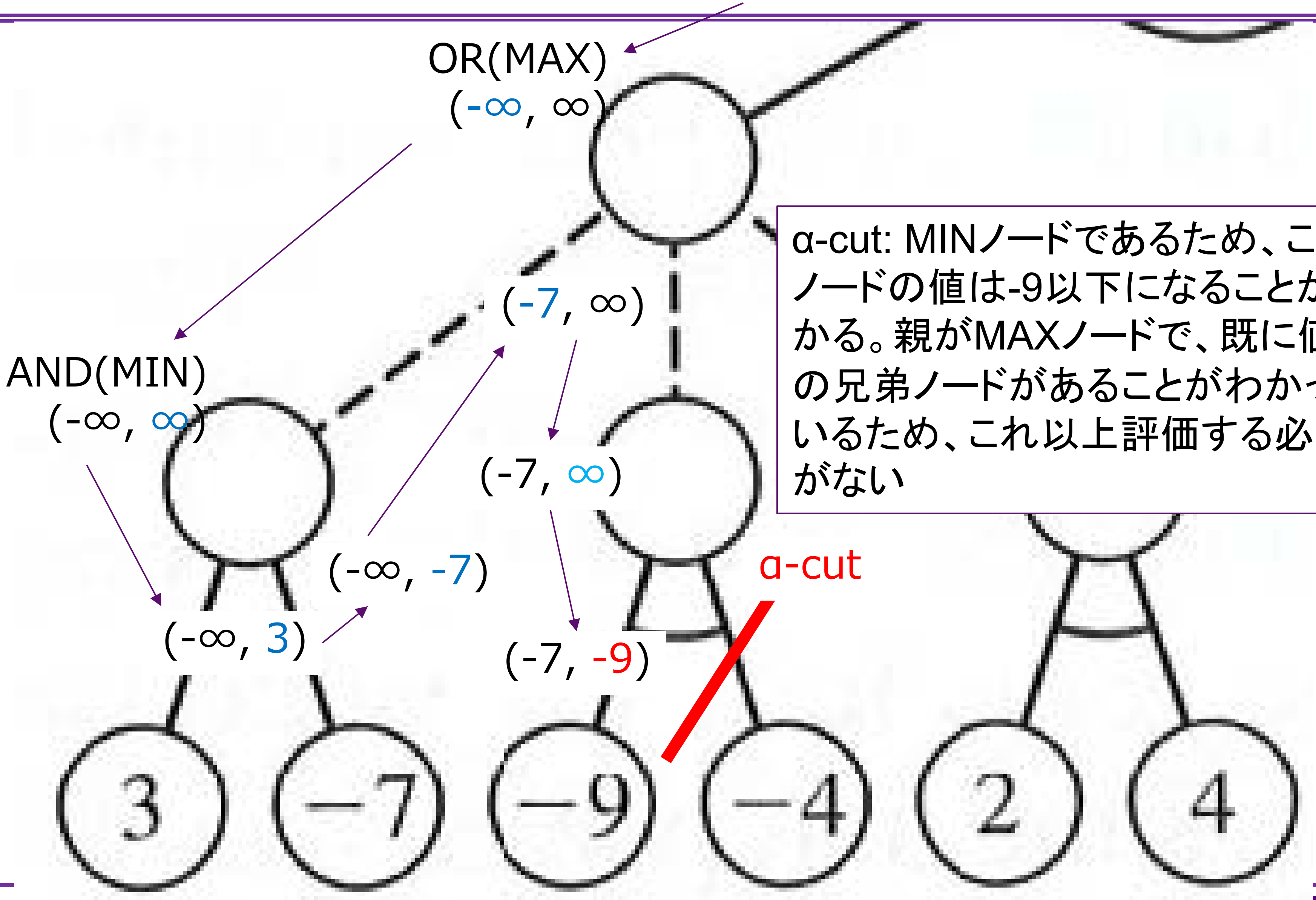
Alpha-Beta枝刈りアルゴリズム

```
function minimax(node, depth)
    return alphabeta(node, depth, -∞, +∞)

function alphabeta(node, depth, α, β)
    if node is leaf or depth = 0
        return value of node
    if node is OR (MAX) node
        foreach child of node
            α = max(α, alphabeta(child, depth-1, α, β))
            if α ≥ β
                return β // β-cut
      return α

    else node is AND (MIN) node
        foreach child of node
            β := min(β, alphabeta(child, depth-1, α, β))
            if α ≥ β
                return α // α-cut
        return β
```

ORノードでは
βより大きい子が
一つでもあれば
残りはcut! (β-cut)

ANDノードでは
αより小さい子が
一つでもあれば
残りはcut! (α-cut)

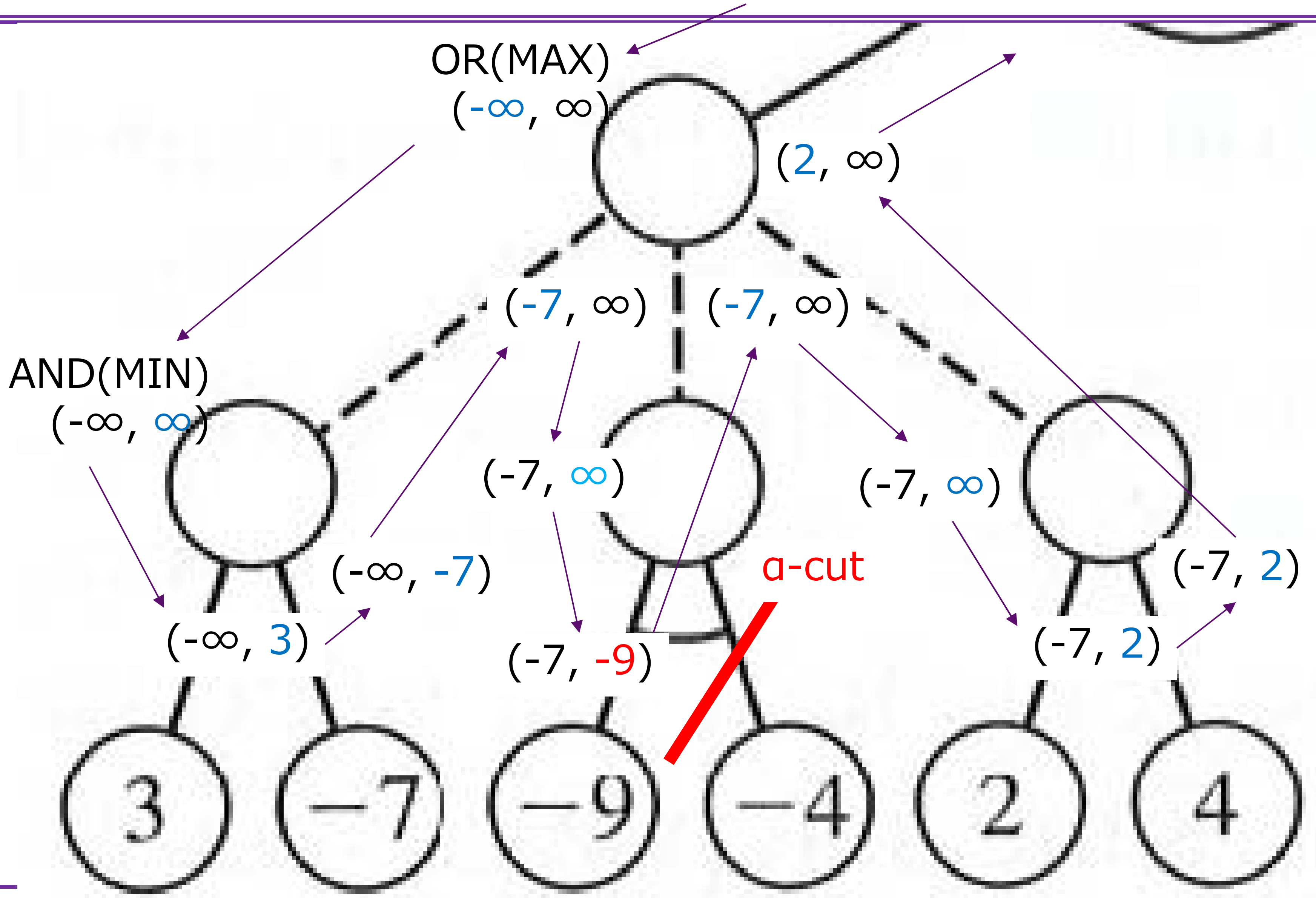
Alpha-Beta枝刈り手法(2)



α-cut: MINノードであるため、このノードの値は-9以下になることがわかる。親がMAXノードで、既に値-7の兄弟ノードがあることがわかっているため、これ以上評価する必要がない

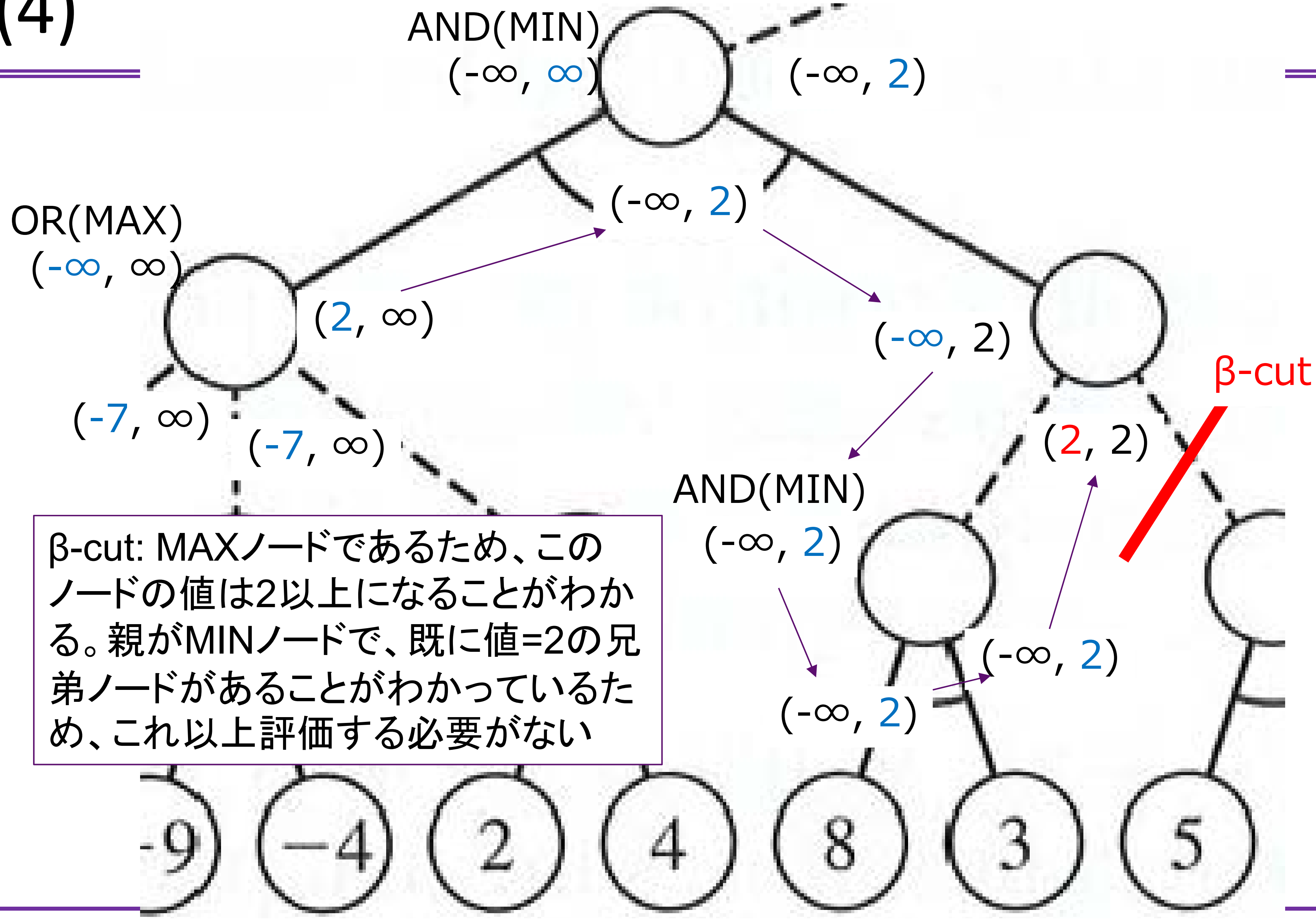
Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

Alpha-Beta枝刈り手法(3)



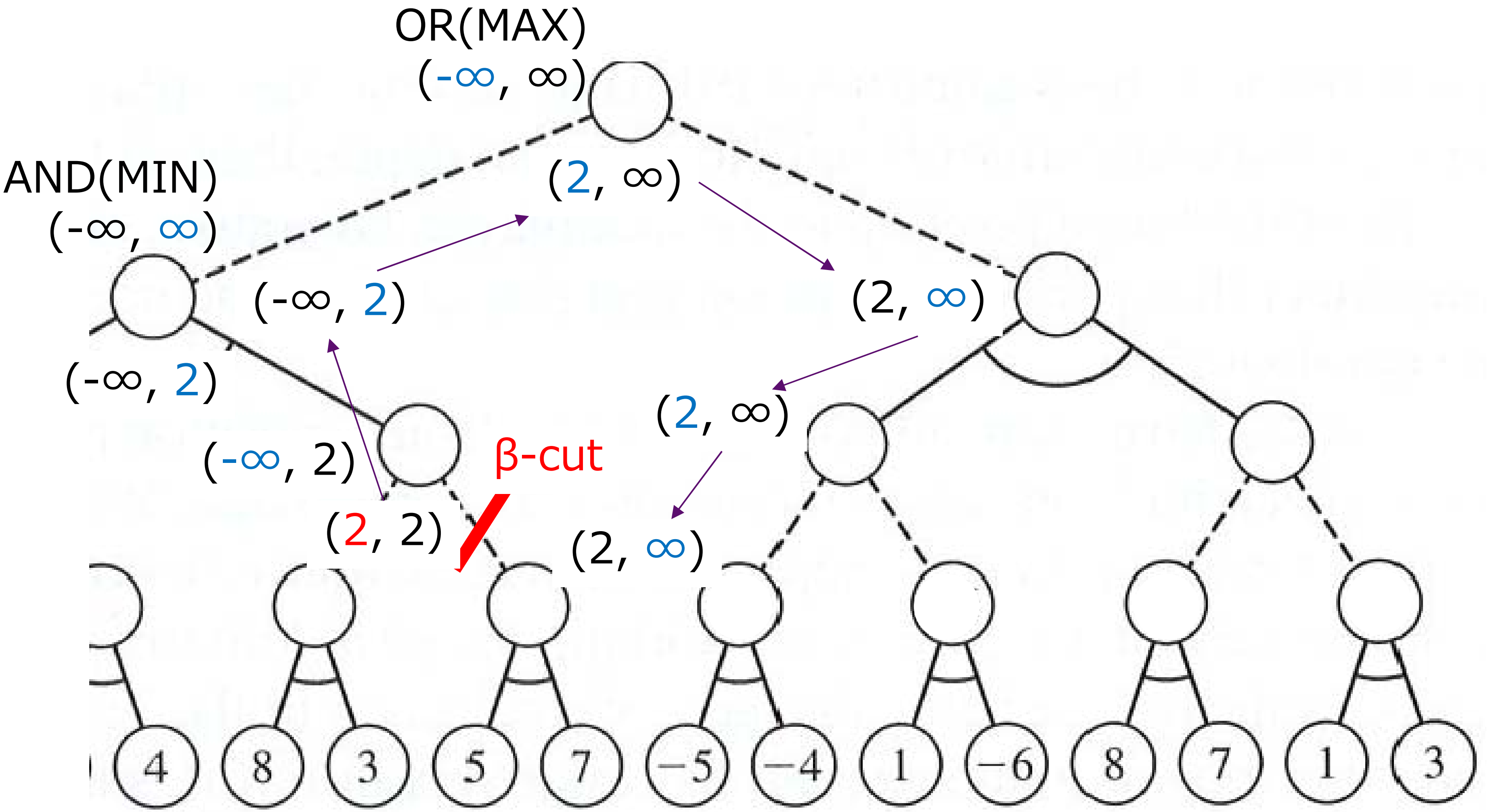
Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

(4)

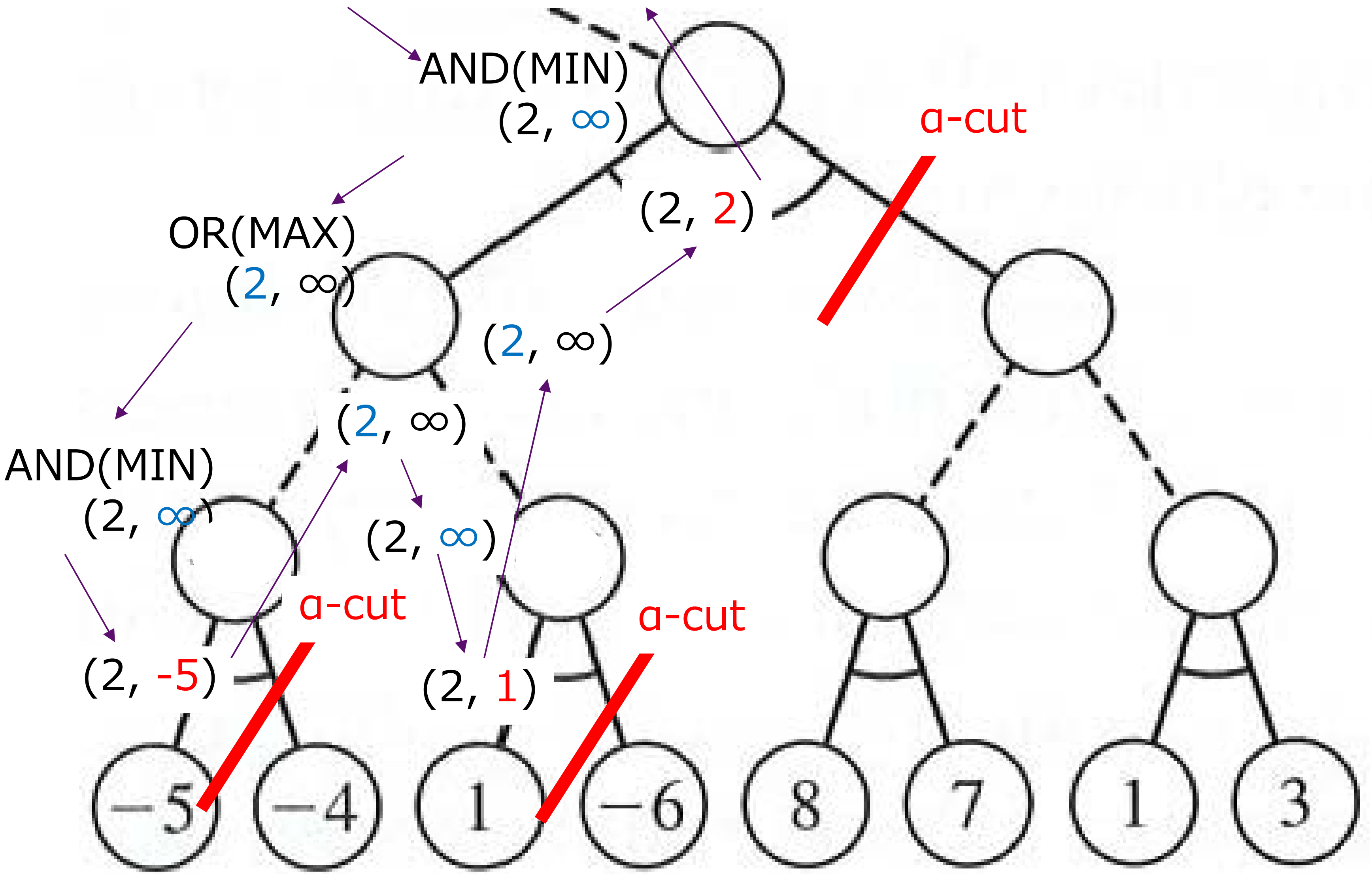


β-cut: MAXノードであるため、このノードの値は2以上になることがわかる。親がMINノードで、既に値=2の兄弟ノードがあることがわかっているため、これ以上評価する必要がない

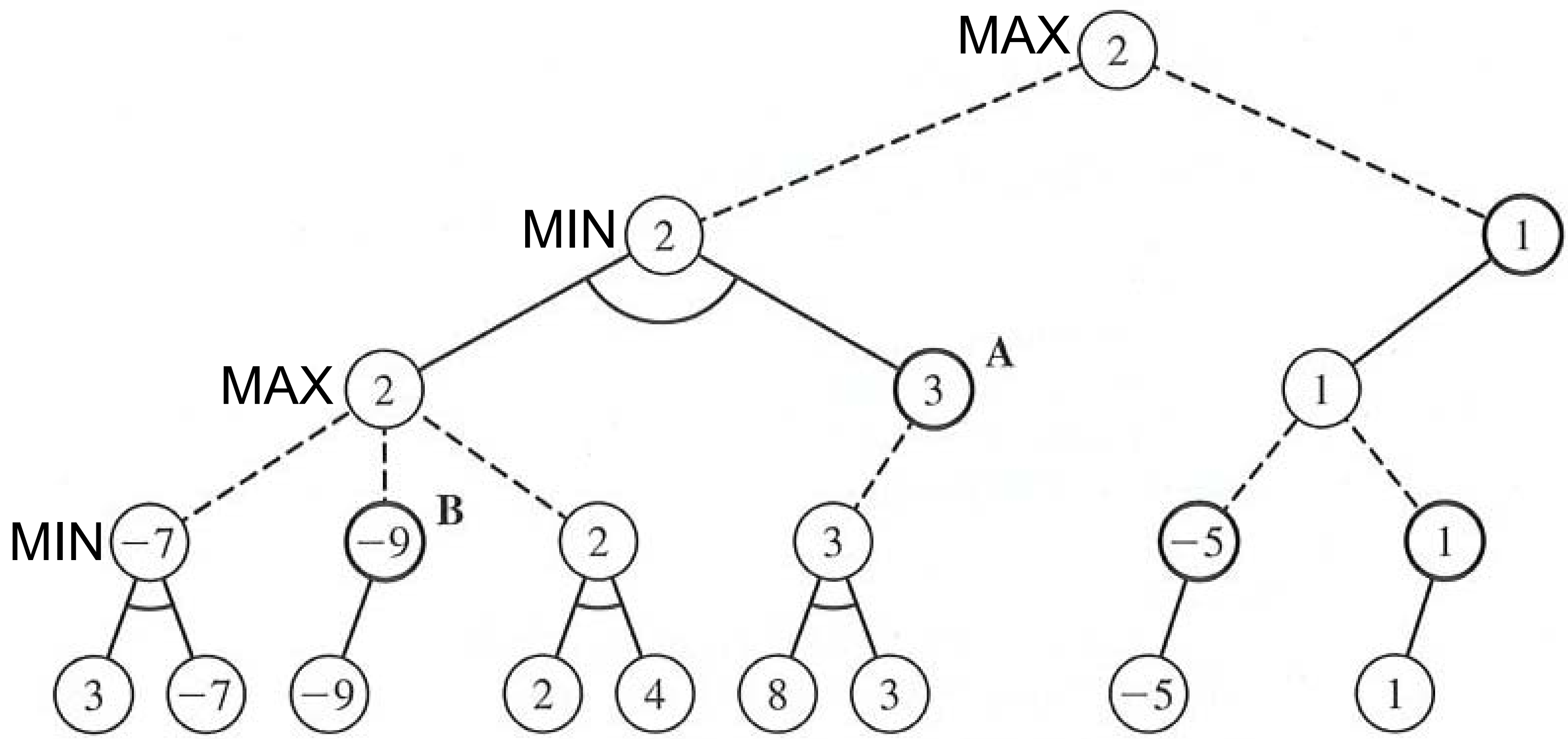
Alpha-Beta枝刈り手法(5)



Alpha-Beta枝刈り手法(6)



Alpha-Beta枝刈り手法(7)

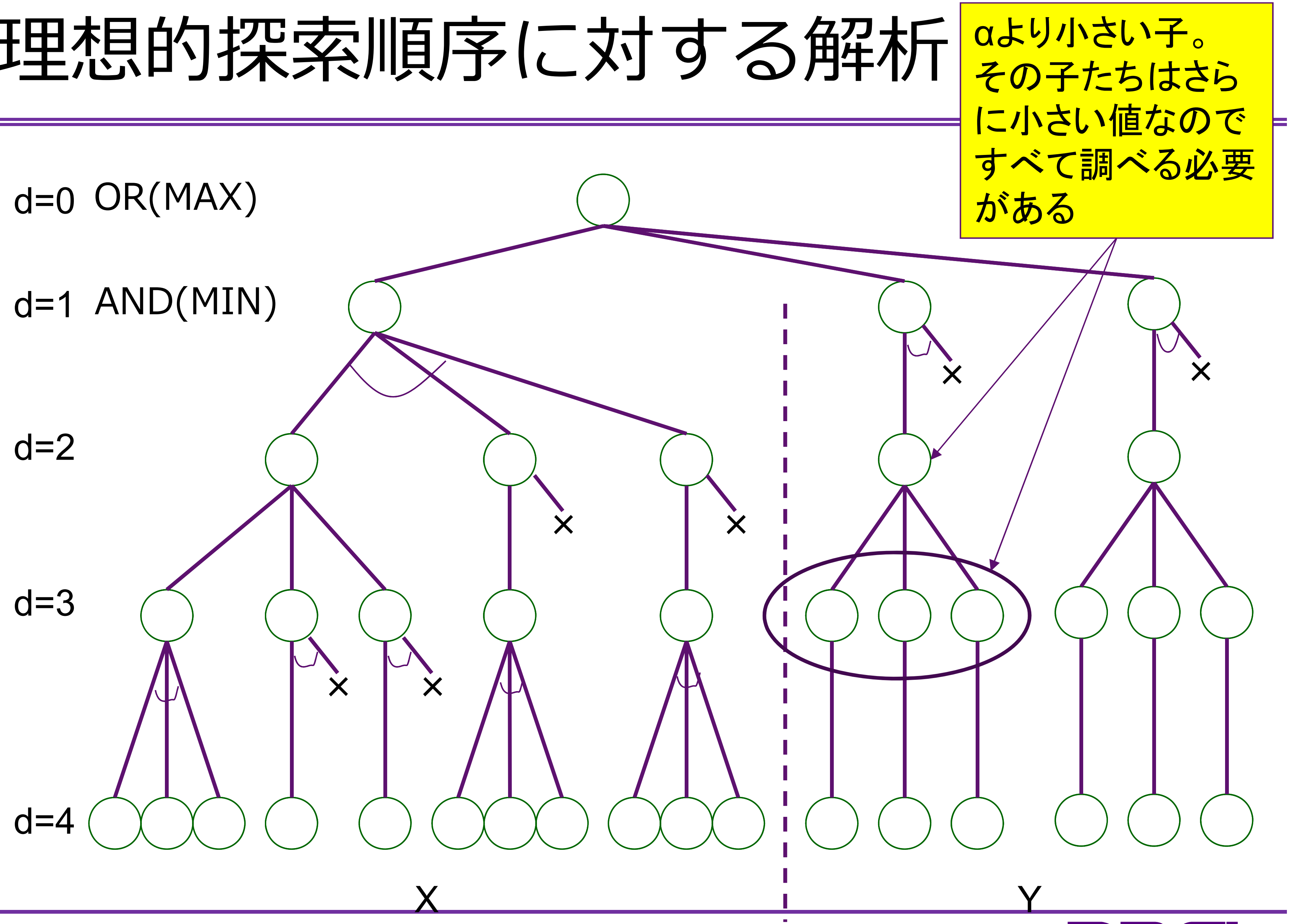


Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

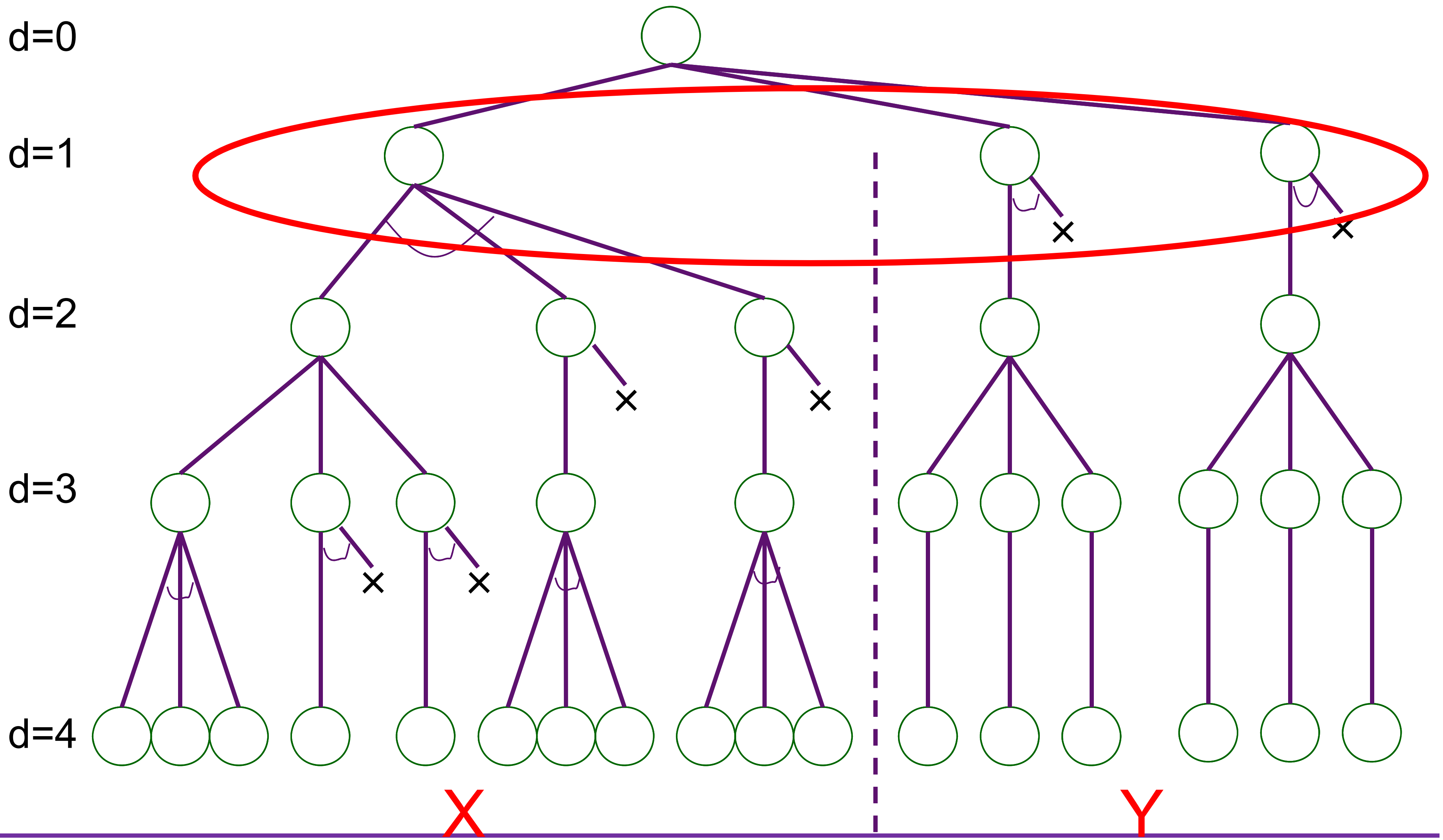
解析

- Slagle and Dixon
  - alpha-betaアルゴリズムにおいて評価される葉（最末端）ノードの数は
    - Opt(b, d) = b^(ceiling(d/2)) + b^(floor(d/2)) – 1
    - b: 木の分岐数, d: 木の深さ
  - J. R. Slagle and J. K. Dixon, “Experiments with Some Programs That Search Game Trees,” J. ACM, Vol. 16, No. 2 (April 1969), pp. 189-207.

理想的探索順序に対する解析

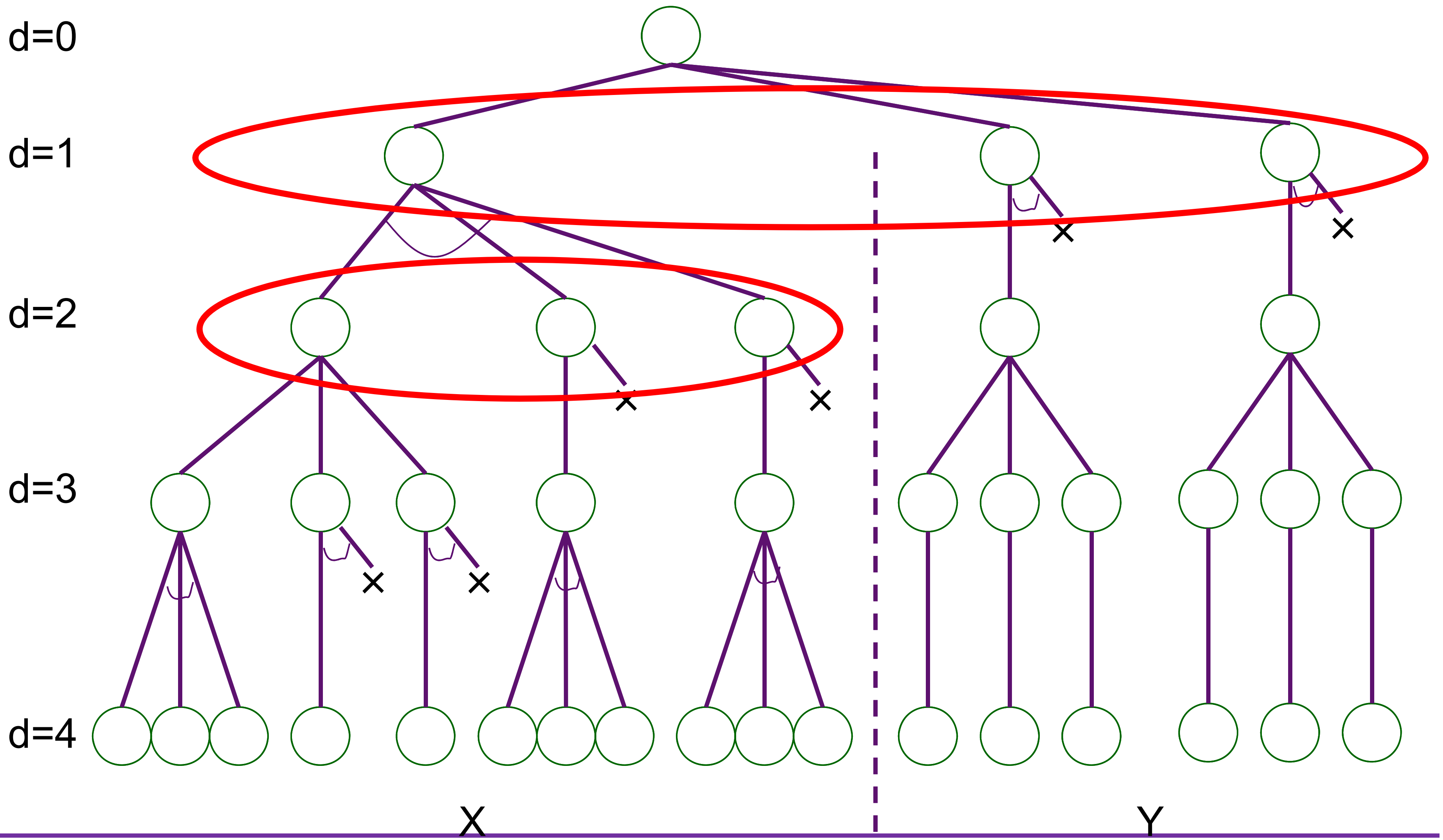


理想的探索順序→最初のノードでα-cut



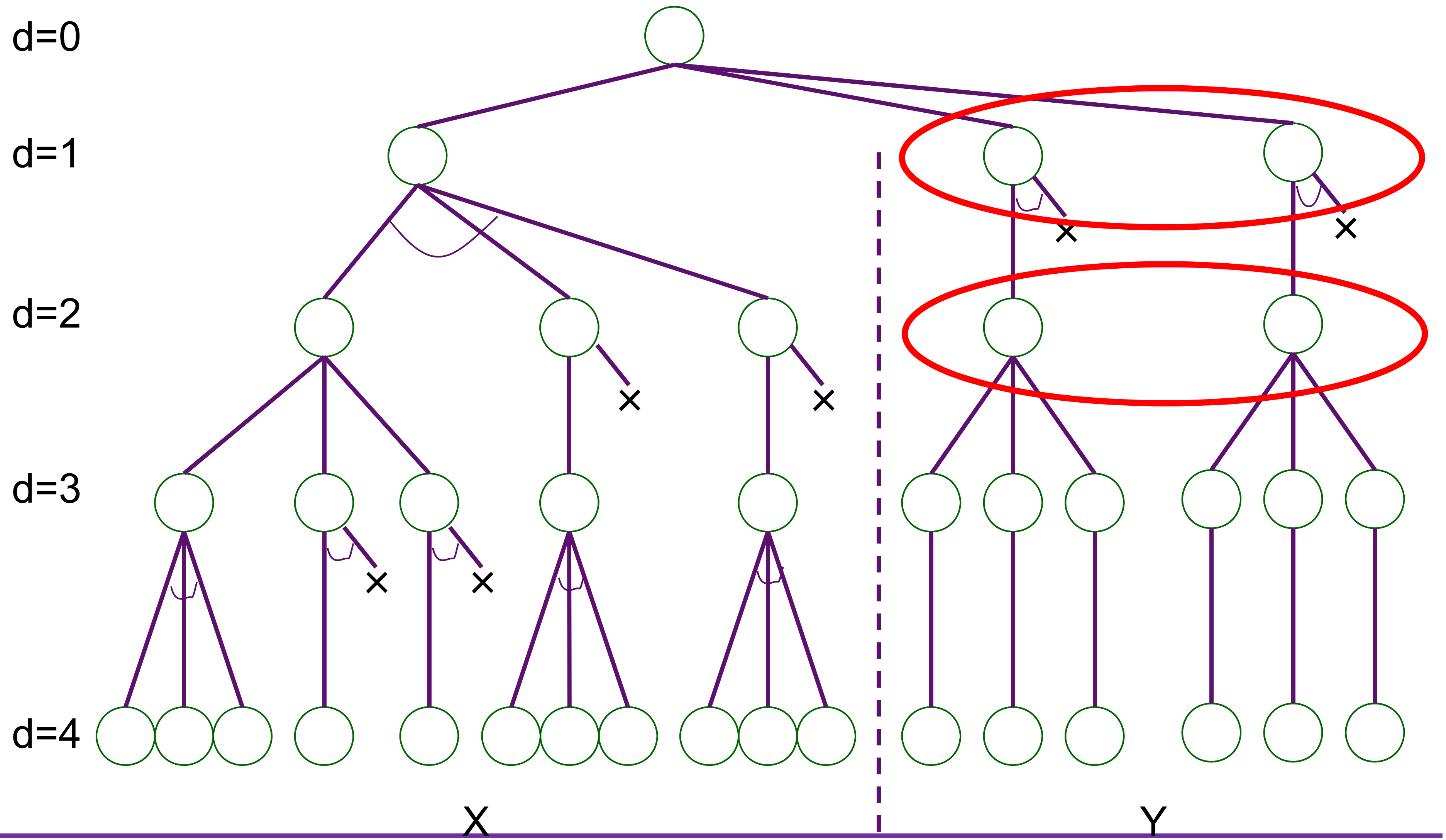
$$X_d = N_{d-1}$$

N_d = 深さdのノード数
X_d = 深さdのX側ノード数
Y_d = 深さdのY側ノード数



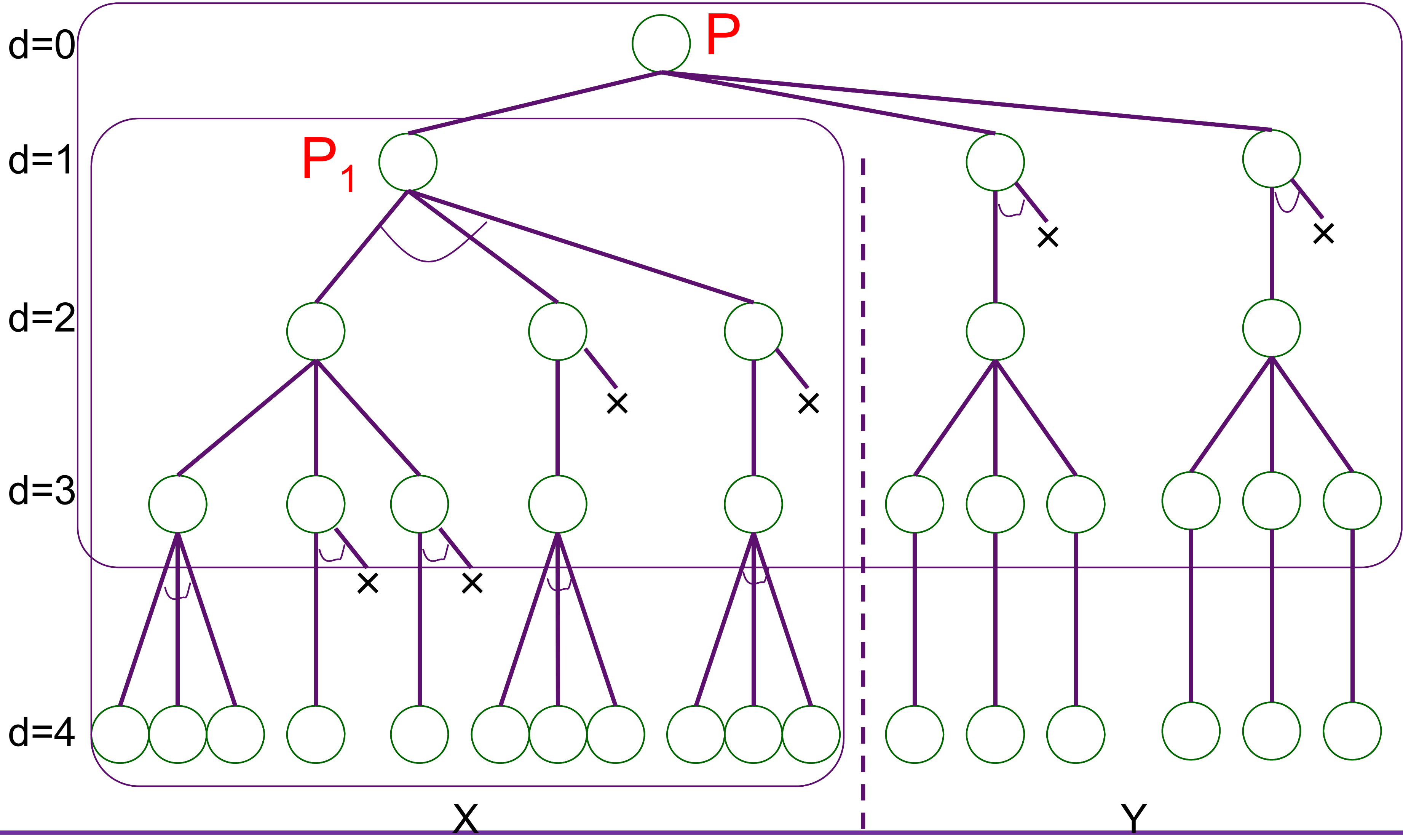
Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

$Y_d = Y_{d-1}$（d：偶数）、$Y_d = B*Y_{d-1}$（d：奇数）



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

P_1から深さdまでのα-β探索は、Pから深さd-1までの探索と同数のノードを評価



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

解析

- Slagle and Dixon
  - alpha-betaアルゴリズムにおいて評価される葉（最末端）ノードの数は
    - Opt(b, d) = b^(ceiling(d/2)) + b^(floor(d/2)) – 1
    - b: 木の分岐数, d: 木の深さ
  - J. R. Slagle and J. K. Dixon, "Experiments with Some Programs That Search Game Trees," J. ACM, Vol. 16, No. 2 (April 1969), pp. 189-207.

高速化

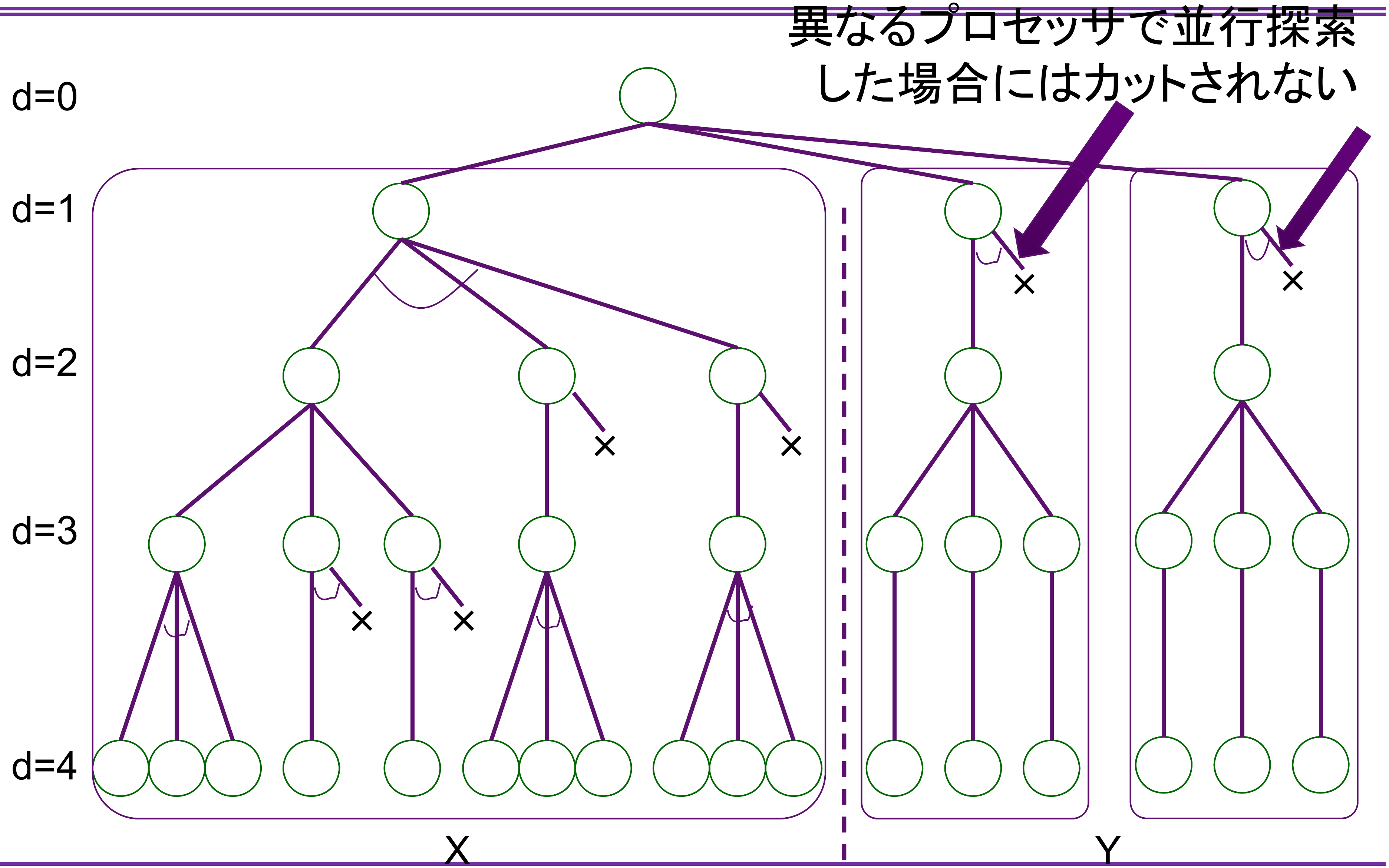
- Aspiration Search （見積値による探索）
  - 見積値 v, 範囲 e
  - 初期値を (α, β) = (v-e, v+e) とする
  - もしもうまくいかなかったら (-∞, v-e) or (v+e, ∞)

- Iterative Deepening（反復深化）
  - 深さ、分岐ともに多い木に対する手法
  - 深さ優先探索: 毎回末端までたどる
  - 幅優先探索、最良値優先探索: メモリ量が大きい
  - 反復深化: 各「探索」において深さを制限、「良い」部分木について継続

並列alpha-beta探索

- 並列Aspiration Search
  - 例えば、３プロセッサあるときに(v-e, v+e), (-∞, v-e), (v+e, ∞)を並列探索
- 並列部分木評価
  - 良さそうに思えるが…
  - Opt(b, d) -> Opt(b, d-1)
    - b=38 (Chess), d=10
    - Opt(b, d) = 158,470,335
    - Opt(b, d-1) = 81,320,303
    - Amdahlの法則の意味でどうなのだろうか？

並列部分木評価



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

証明数と反証数

- 各ノードの評価値を決める手法の一つ
- 証明数 pn(n): あるノードが“true (win)” であることを証明するために評価すべき最小ノード数
- 反証数 dn(n):あるノードが“false (lose)” であることを証明するために評価すべき最小ノード数

- 証明数もしくは反証数が小さいようなノードに向けて探索を進める

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

証明数と反証数

- 葉ノード n に対して
  - ノード n が trueの場合: pn(n)=0, dn(n)=∞
  - ノード n が falseの場合: pn(n)=∞, dn(n)=0
  - その他の場合 (わからない場合): pn(n)=1, dn(n)=1

- 中間ノード n に対して
  - ORノード (MAXノード):
    pn(n)=MIN_child pn(child)
    dn(n)=Σ_child dn(child)
  - ANDノード (MINノード):
    pn(n)=Σ_child pn(child)
    dn(n)=MIN_child dn(child)

並列探索

- しばしば、証明数、反証数が共に多い部分木に最適解がある

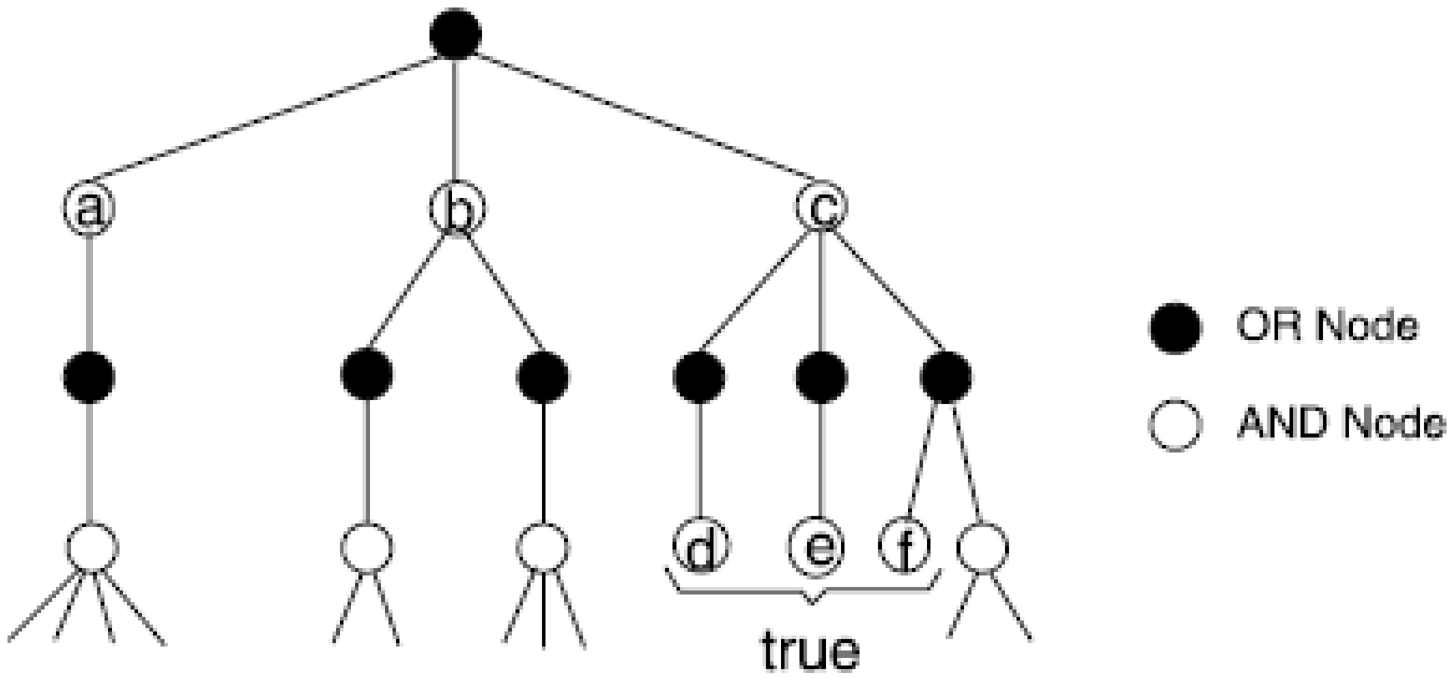


図 1　探索木の例
Fig. 1　An example of search tree.

階層的挟み撃ち法

- 一つのプロセッサは、評価値の良いノードに向けて探索を進める
- 他のプロセッサは、上記探索経路の「兄弟ノード」を探索する

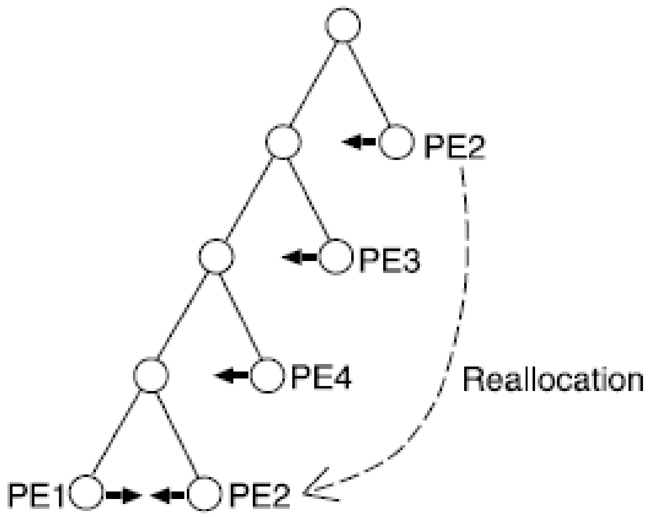


図 2　階層的挟み撃ち探索

Fig. 2　Hierarchical pincers attack search.

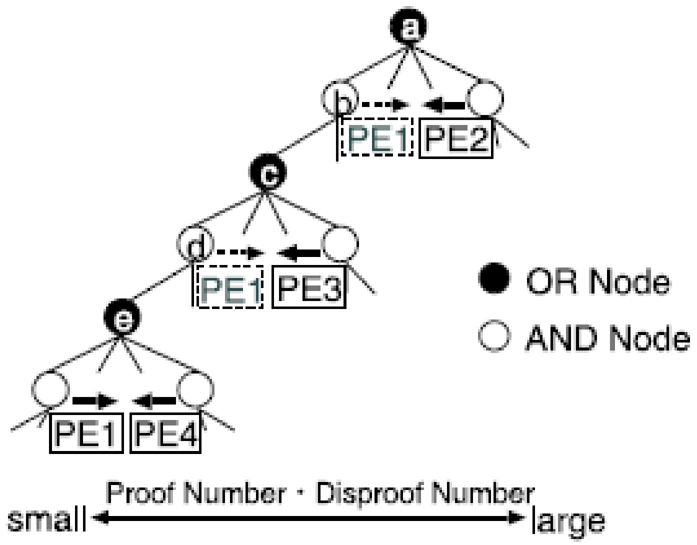


図 4　AOHPAS の探索例

Fig. 4　An example of search of AOHPAS.

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

評価結果(1)

- もとのアルゴリズムにおいてうまくいかないデータに対し、大きな効果

表 2 PDS での探索時間別の高速化率

Table 2 Speedups by range of search time using PDS.

Range of Sequential Search Time t[s]	Number of Problems	avg.	max.	min.
$0 < t < 1$	3	0.17	0.37	0.07
$1 \leq t < 10$	4	0.33	1.27	0.03
$10 \leq t < 100$	5	1.59	13.12	0.52
$100 \leq t < 1000$	4	2.40	6.86	0.58
$1000 \leq t < 3600$	1	140.96	140.96	140.96
$3600 \leq t$	10	15.09	214.93	1.57

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

評価結果(2)

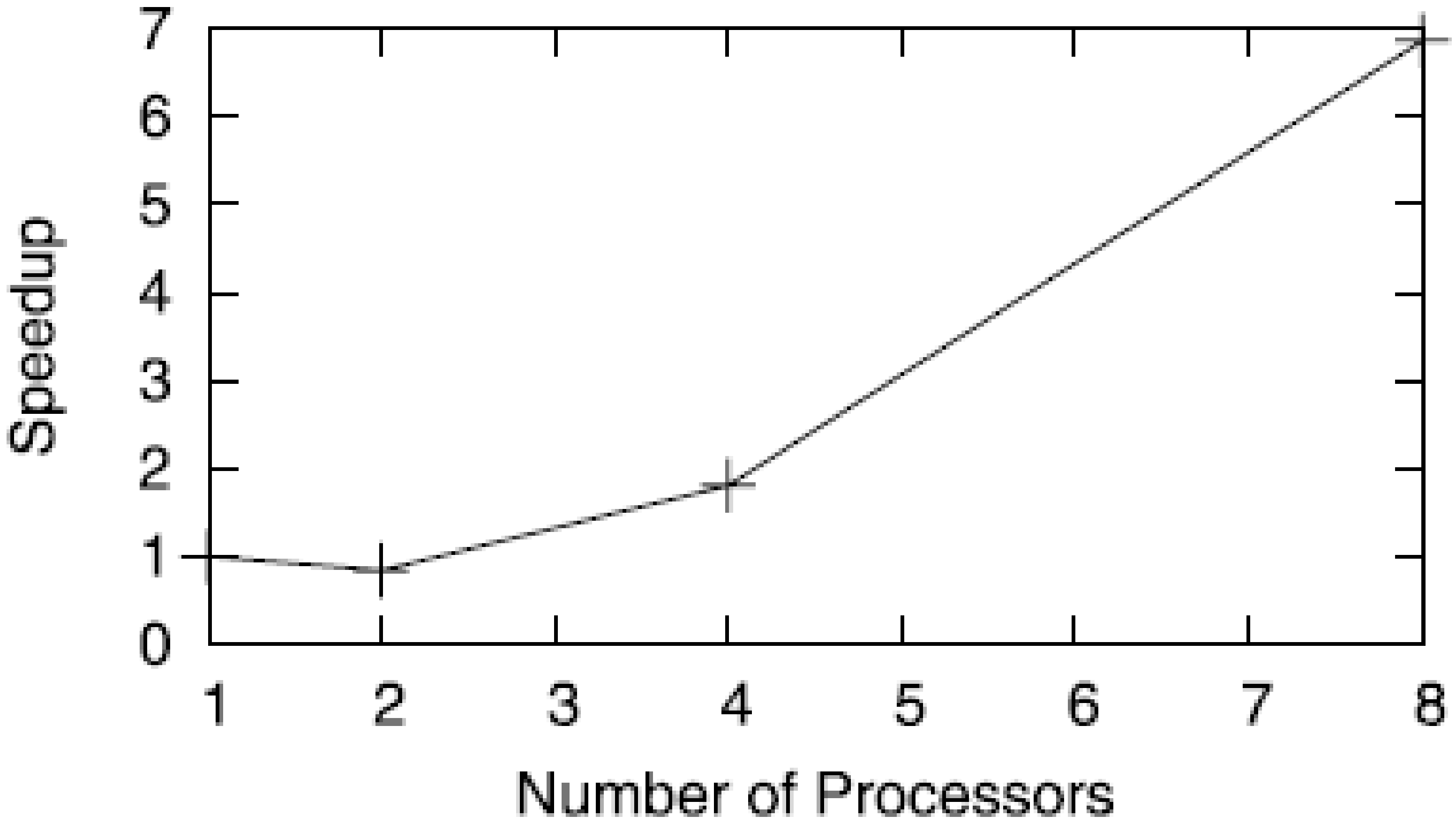


図 10　プロセッサ数以下の高速化が見られる例
Fig. 10　An example of under linear speedup.

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

評価結果(3)

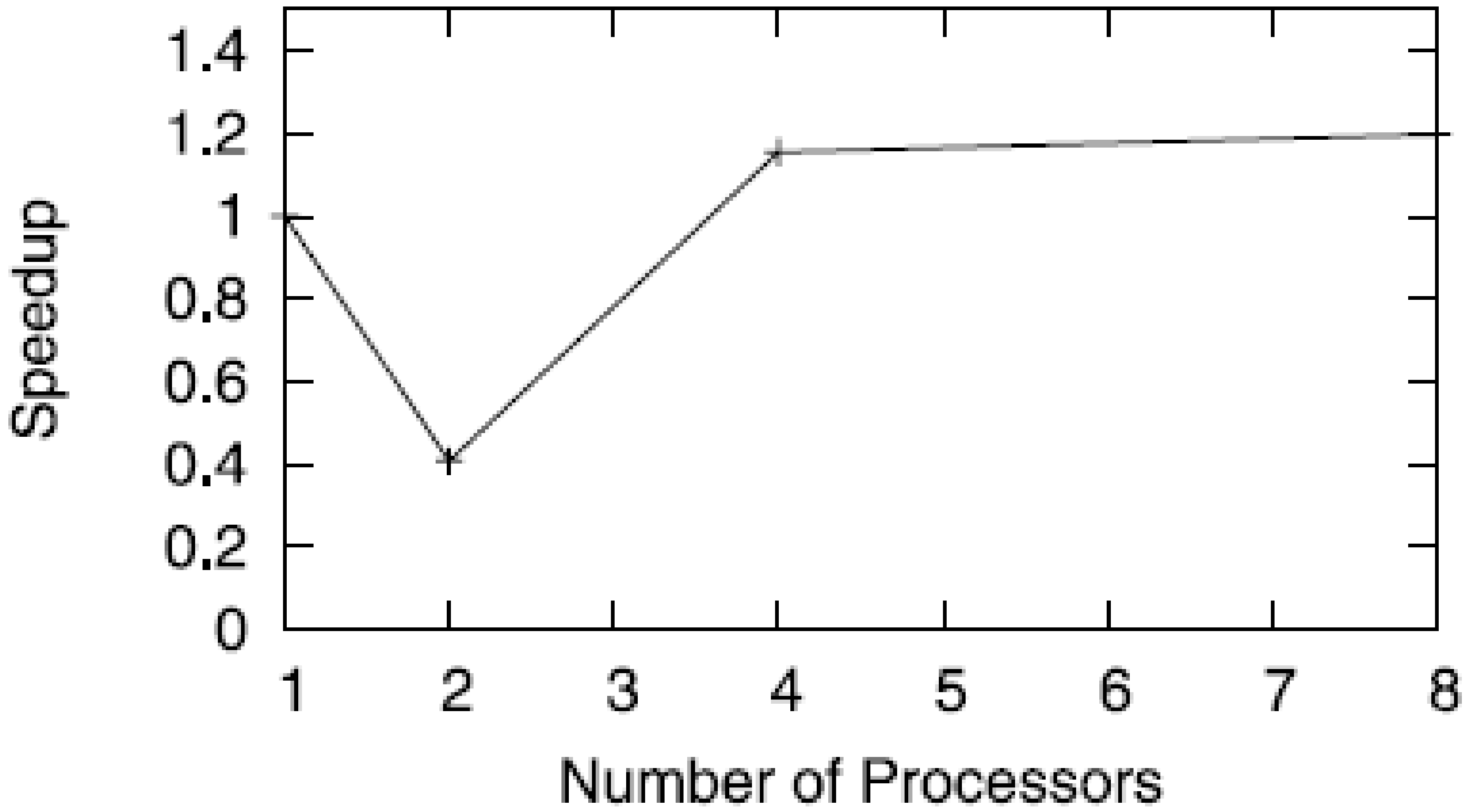


図 9　高速化が得られない例

Fig. 9　An example of no speedup.

評価結果(4)

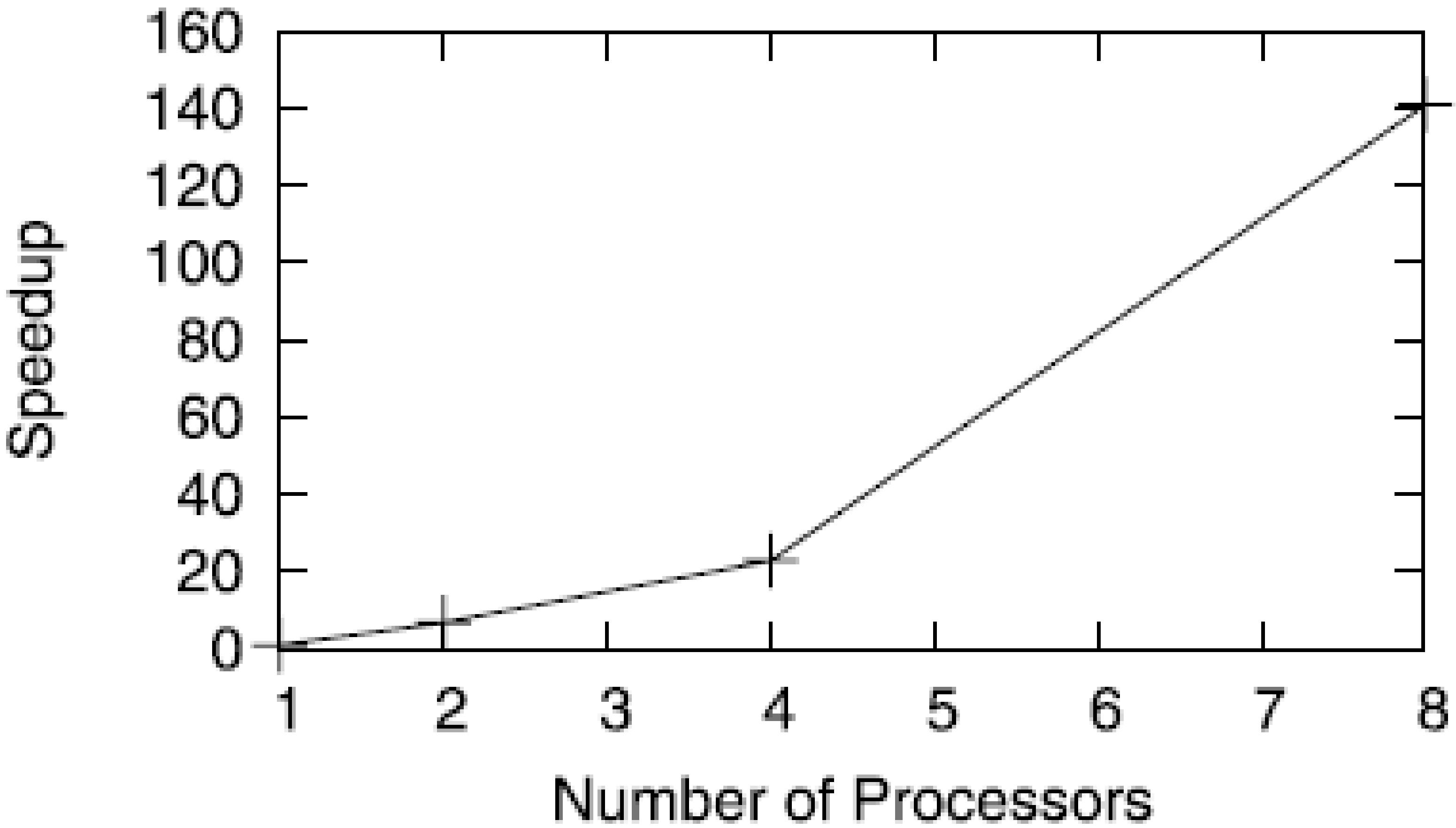


図 11　プロセッサ数以上の高速化が得られる例
Fig. 11　An example of super linear speedup.

演習3

- 並列プログラミング または 並列アルゴリズムに関するレポート
  - 並列プログラミング
    - 次のページ
  - レポート課題
    - 10頁程度の翻訳
- すべての提出物(プログラム，レポート，理解度テスト，アンケート)は8月21日までにeda@ertl.jpに送ること

演習3: 並列アルゴリズム

- 問題を自由に選択し、２種類のアルゴリズム（少なくとも一種は並列であること）で実装を行い、並列アルゴリズムのスケーラビリティを評価せよ(〆切：８月２1日: August 21, to eda@ertl.jp)
  1. 問題を自由に選択
  2. 同じ入出力になるはずの２種類のアルゴリズム（少なくとも一種は並列）を実装せよ。例えば:
     - 並列ソートやparallel prefixに関し、単純な逐次アルゴリズムと、巧妙な並列アルゴリズム
  3. レポートを提出
     1. ２種のアルゴリズムで結果の一致性を確認
     2. アムダールの法則の意味での並列可能部分の割合を算出し、並列アルゴリズムのスケーラビリティを評価せよ

演習3: レポート課題

- 本の一部または論文を10頁程度翻訳する
  - 英語から母国語（英語を除く）
  - 日本語から母国語（日本語を除く）
  - レポート課題を選択する場合，希望する言語と共に7月末までに枝廣に連絡すること。それに応じて翻訳すべき課題を連絡する