Discrete Strategy Improvement for Solving Parity Games

Oliver Friedmann

Department of Computer Science,
Ludwig-Maximilians-Universität Munich, Germany.

February 15, 2011

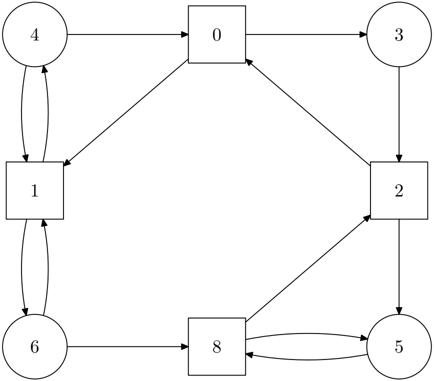
Parity Games

Parity games

Parity Game (PG) $G = (V, V_0, V_1, E, \Omega)$.

- Two players called 0 and 1, with $V = V_0 \cup V_1$ (and $V_0 \cap V_1 = \emptyset$)
- Labeling as a priority assignment $\Omega : V \to \mathbb{N}$
- Winning objective: player 0 wins if largest priority seen infinitely often is even

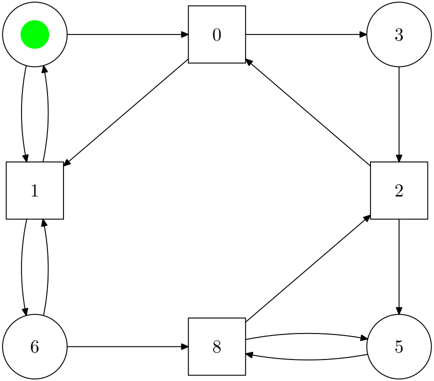
Example



} player 0 controlled

} player 1 controlled

Example

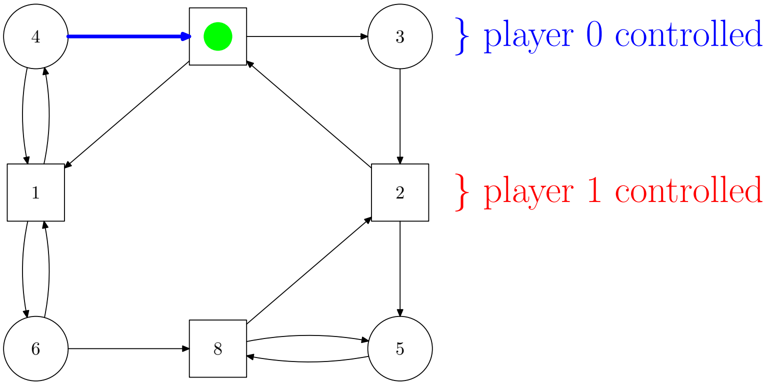


} player 0 controlled

} player 1 controlled

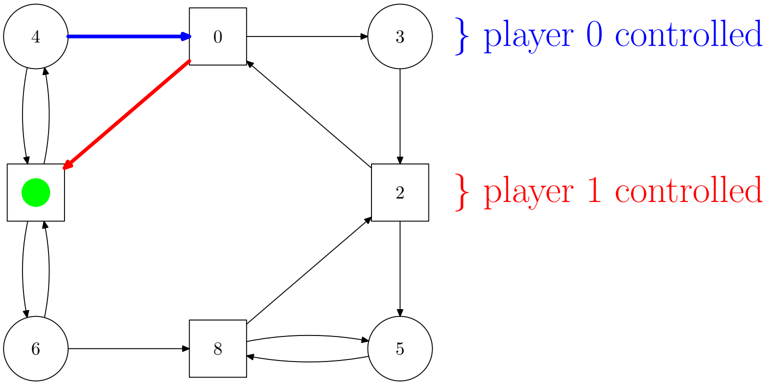
Play: 4,

Example



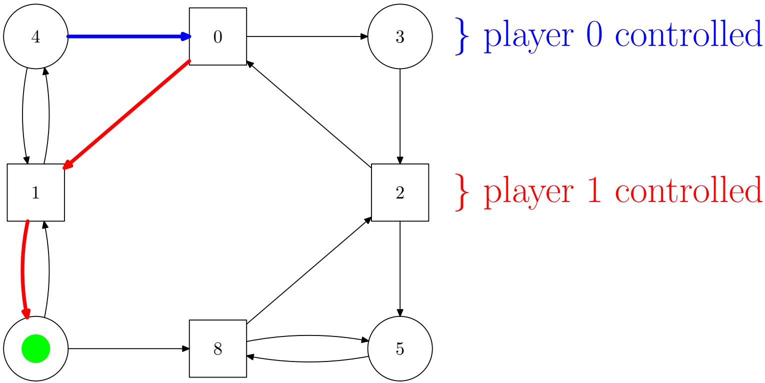
Play: 4, 0,

Example



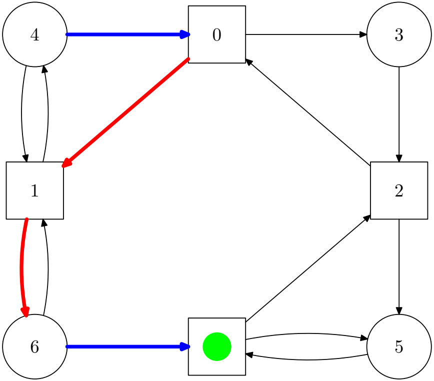
Play: 4, 0, 1,

Example



Play: 4, 0, 1, 6,

Example

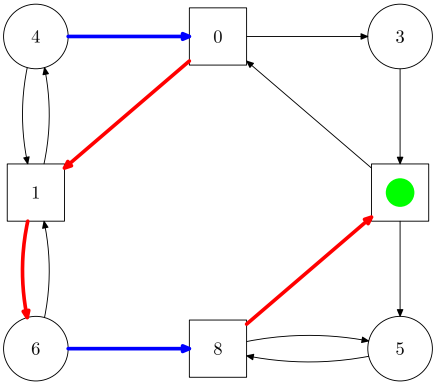


} player 0 controlled

} player 1 controlled

Play: 4, 0, 1, 6, 8,

Example

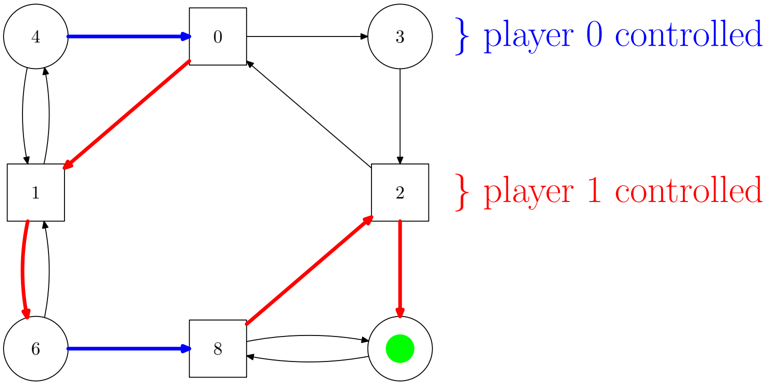


} player 0 controlled

} player 1 controlled

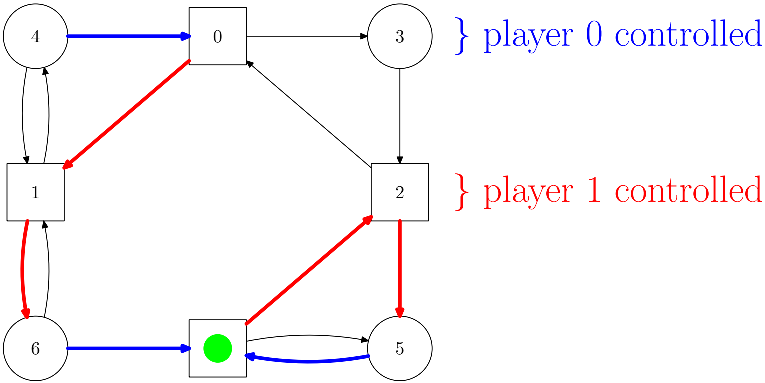
Play: 4, 0, 1, 6, 8, 2,

Example



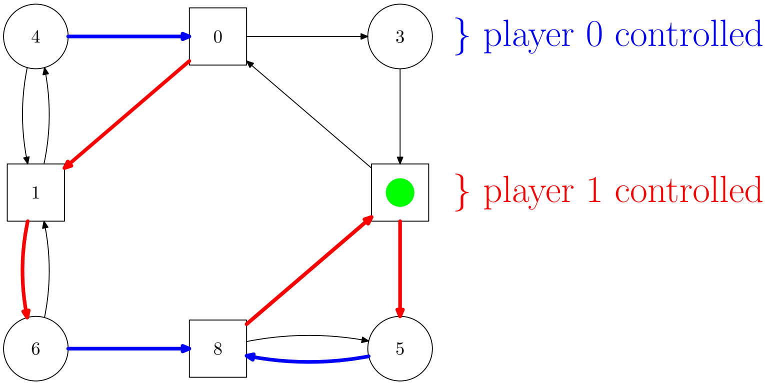
Play: 4, 0, 1, 6, 8, 2, 5,

Example



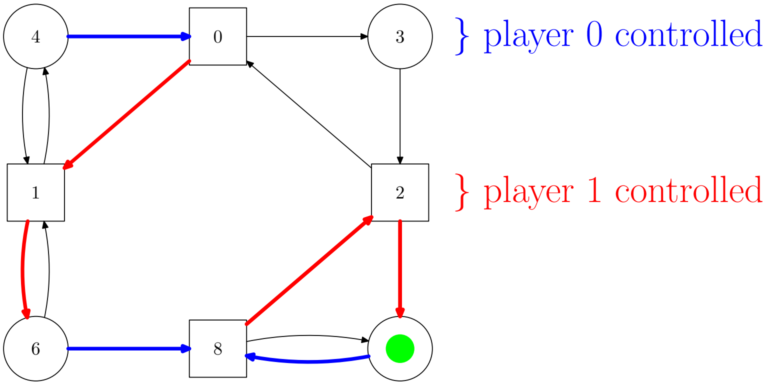
Play: 4, 0, 1, 6, 8, 2, 5, 8,

Example



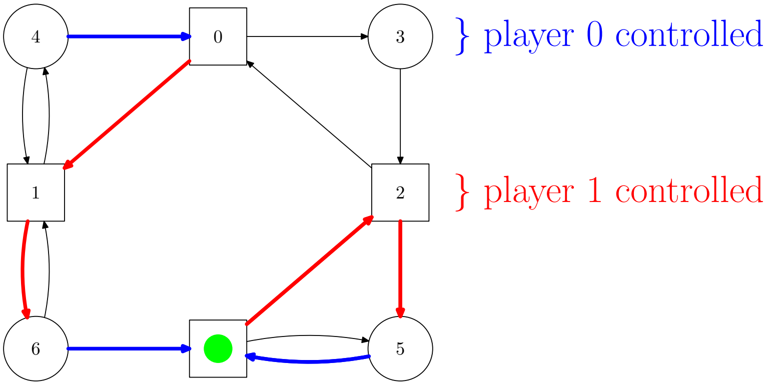
Play: 4, 0, 1, 6, 8, 2, 5, 8, 2,

Example



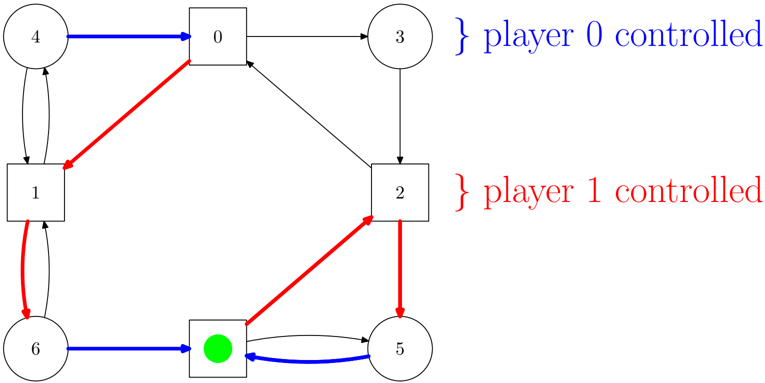
Play: 4, 0, 1, 6, 8, 2, 5, 8, 2, 5,

Example



Play: 4, 0, 1, 6, 8, 2, 5, 8, 2, 5, 8, ...

Example



Play: 4, 0, 1, 6, $(8, 2, 5)^\omega$

Winner is player 0: largest priority seen infinitely often, 8, is even

Strategies, Winning, and all that

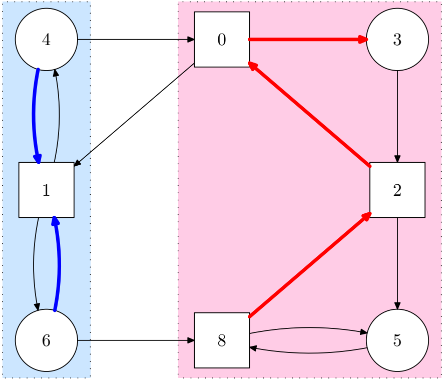
- (Positional) strategy for player i: $\sigma_i : V_i \to V$ respecting E
- v-winning strategy σ for i: wins v-starting plays against any counterstrategy
- Winning set $W_i = \{$ nodes for which i has a winning strategy $\}$

Theorem

The set of vertices V can be partitioned into winning sets W_0 and W_1. Player i has a single positional winning strategy for all nodes in W_i.

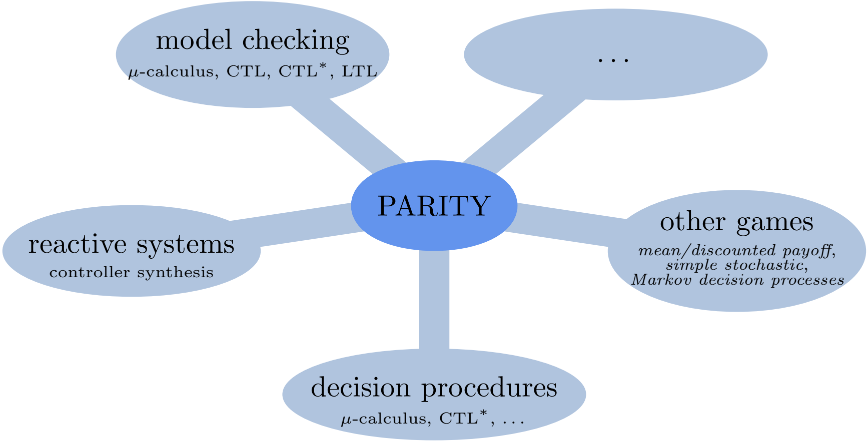
Computational Problem: Compute W_0 and W_1 along with winning strategies.

Example



Player 0 wins blue area with blue strategy.
Player 1 wins red area with red strategy.

Applications



Modal Logic Decision Procedures

Validity Problem for Modal μ-calculus

$$\models \varphi \,?$$

Modal Logic Decision Procedures

Validity Problem for Modal μ-calculus

$$\models \varphi \; ?$$

Infinite Logic Tableau labeled with subformulas and local correctness conditions

Modal Logic Decision Procedures

Validity Problem for Modal μ-calculus

$$\models \varphi\ ?$$

Nondeterministic Büchi Automaton on subformulas of infinite branches

Infinite Logic Tableau labeled with subformulas and local correctness conditions

Modal Logic Decision Procedures

Validity Problem for Modal μ-calculus

$$\models \varphi \;?$$

Nondeterministic Büchi Automaton on subformulas of infinite branches

$\downarrow$

Deterministic Parity Automaton on subformulas of infinite branches

Infinite Logic Tableau labeled with subformulas and local correctness conditions

Modal Logic Decision Procedures

Validity Problem for Modal μ-calculus

$\models \varphi$?

Nondeterministic Büchi Automaton on subformulas of infinite branches

Deterministic Parity Automaton on subformulas of infinite branches

Infinite Logic Tableau labeled with subformulas and local correctness conditions

Winner of Parity Game determines validity of φ

Complexity

Theorem

Parity Games solving is in $\mathtt{NP} \cap \mathtt{coNP}$, $\mathtt{UP} \cap \mathtt{coUP}$ *and* $\mathtt{PLS}$.

Complexity

Theorem

Parity Games solving is in $\mathtt{NP} \cap \mathtt{coNP}$, $\mathtt{UP} \cap \mathtt{coUP}$ *and* $\mathtt{PLS}$.

But: neither known to be in $\mathtt{P}$ nor known to be $\mathtt{NP}$-hard

Complexity

Theorem

Parity Games solving is in $\mathtt{NP} \cap \mathtt{coNP}$, $\mathtt{UP} \cap \mathtt{coUP}$ *and* $\mathtt{PLS}$.

But: neither known to be in $\mathtt{P}$ nor known to be $\mathtt{NP}$-hard

Note: Not many other (natural) problems with similar status!

Complexity

Theorem

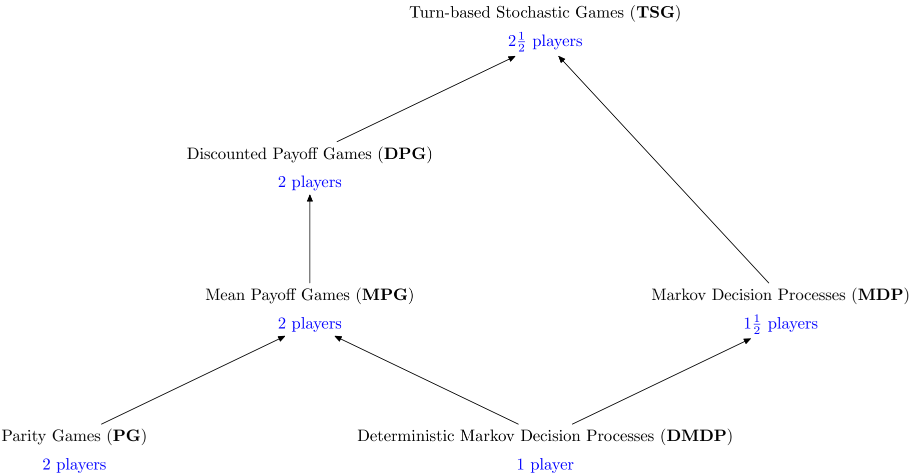
Parity Games solving is in $\mathtt{NP} \cap \mathtt{coNP}$, $\mathtt{UP} \cap \mathtt{coUP}$ *and* $\mathtt{PLS}$.

But: neither known to be in $\mathtt{P}$ nor known to be $\mathtt{NP}$-hard

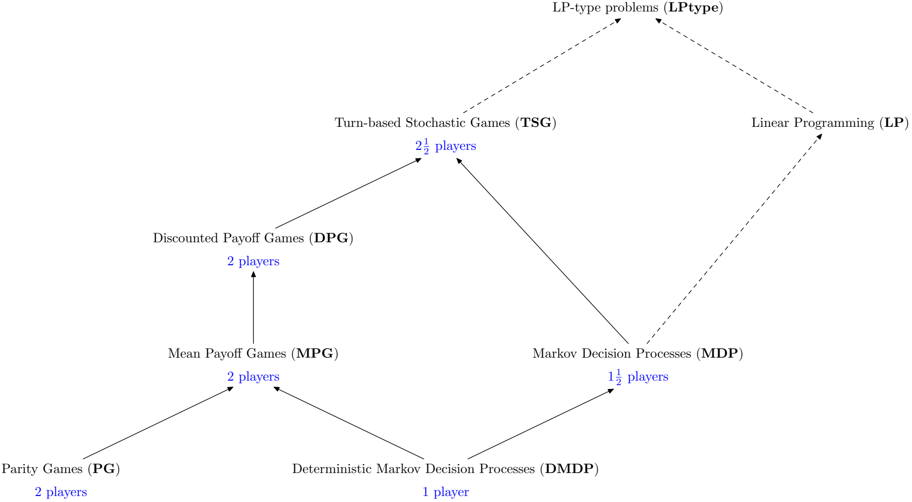
Note: Not many other (natural) problems with similar status!

- Graph isomorphism problem
- Factorization problem

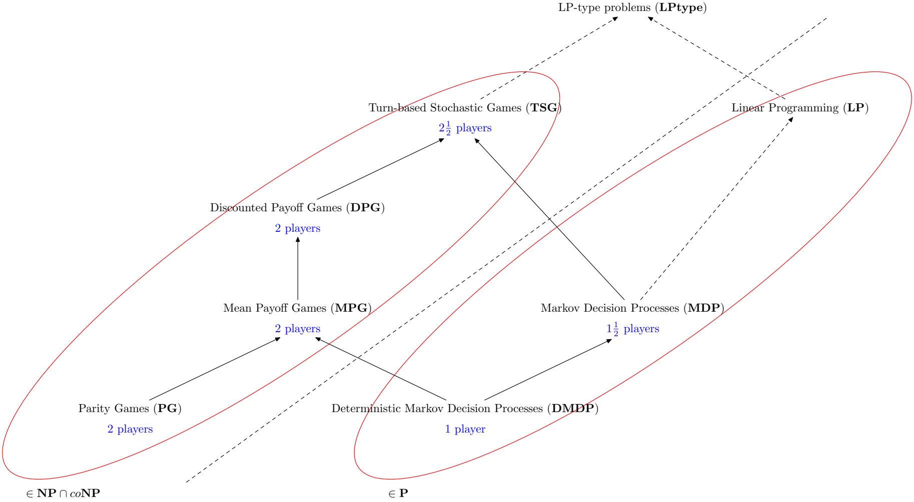
Overview


Turn-based Stochastic Games (**TSG**)
$2\frac{1}{2}$ players
Discounted Payoff Games (**DPG**)
2 players
Mean Payoff Games (**MPG**)
2 players
Markov Decision Processes (**MDP**)
$1\frac{1}{2}$ players
Parity Games (**PG**)
2 players
Deterministic Markov Decision Processes (**DMDP**)
1 player

Overview


LP-type problems (**LPtype**)
Turn-based Stochastic Games (**TSG**)
$2\frac{1}{2}$ players
Linear Programming (**LP**)
Discounted Payoff Games (**DPG**)
2 players
Mean Payoff Games (**MPG**)
2 players
Markov Decision Processes (**MDP**)
$1\frac{1}{2}$ players
Parity Games (**PG**)
2 players
Deterministic Markov Decision Processes (**DMDP**)
1 player

Overview



two players make our life really difficult!

Algorithms

Overview

Algorithms

Overview

1. Recursive Algorithm due to Zielonka
   exponential lower bound

Algorithms

Overview

1. Recursive Algorithm due to Zielonka
   exponential lower bound

2. Small Progress Measures Algorithm due to Jurdziński
   exponential lower bound

Algorithms

Overview

1. Recursive Algorithm due to Zielonka
   exponential lower bound

2. Small Progress Measures Algorithm due to Jurdziński
   exponential lower bound

3. μ-calculus Model Checking Algorithm due to Stevens and Stirling
   exponential lower bound

Algorithms

Overview

1. Recursive Algorithm due to Zielonka
   exponential lower bound
2. Small Progress Measures Algorithm due to Jurdziński
   exponential lower bound
3. μ-calculus Model Checking Algorithm due to Stevens and Stirling
   exponential lower bound
4. Strategy Improvement Algorithm due to Vöge and Jurdziński
   exponential lower bounds but linear diameter

Algorithms

Overview

1. Recursive Algorithm due to Zielonka
   exponential lower bound
2. Small Progress Measures Algorithm due to Jurdziński
   exponential lower bound
3. μ-calculus Model Checking Algorithm due to Stevens and Stirling
   exponential lower bound
4. Strategy Improvement Algorithm due to Vöge and Jurdziński
   exponential lower bounds but linear diameter

Strategy Improvement applies to other infinitary payoff games as well!

Implementations

1. Parity Game Solver Platform
   PGSolver: http://www.tcs.ifi.lmu.de/pgsolver

2. Modal Logic Solver Platform
   MLSolver: http://www.tcs.ifi.lmu.de/mlsolver

Strategy Improvement

History

- Howard 1960: infinite-horizon Markov Decision Processes
- Hoffman-Karp 1966: Non-terminating Stochastic Games
- Condon 1992: Simple Stochastic Games
- Puri 1995: Discounted Payoff Games
- Vöge-Jurdziński 2000: discrete algorithm to solve Parity Games

Scheme

Two ingredients...

Scheme

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies

Scheme

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$

Scheme

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$

with two properties...

Scheme

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$

with two properties...

1. $\trianglelefteq$-optimal strategy σ induces winning sets and winning strategies

Scheme

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$

with two properties...

1. $\trianglelefteq$-optimal strategy σ induces winning sets and winning strategies
2. σ not $\trianglelefteq$-optimal implies $\sigma \lhd \textsc{Improve}(\sigma)$

Scheme

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$

with two properties...

1. $\trianglelefteq$-optimal strategy σ induces winning sets and winning strategies
2. σ not $\trianglelefteq$-optimal implies $\sigma \lhd \textsc{Improve}(\sigma)$

Strategy Improvement

1: **while** σ is not optimal **do**
2: $\quad \sigma \leftarrow \textsc{Improve}(\sigma)$
3: **end while**

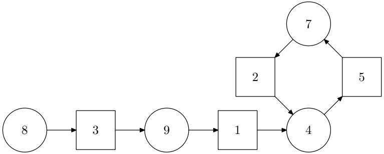
Question: does this algorithm always terminate?

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

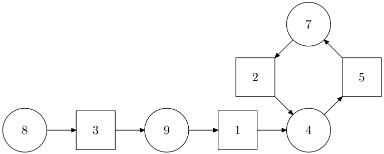


Positional Play

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

$$\pi_{\sigma,\tau,v} \mapsto \mathrm{val}_{\sigma,\tau}(v)$$

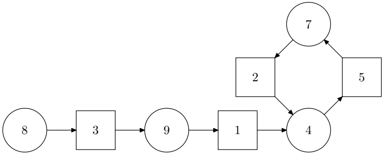


Positional Play

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

$$\pi_{\sigma,\tau,v} \mapsto \mathrm{val}_{\sigma,\tau}(v)$$



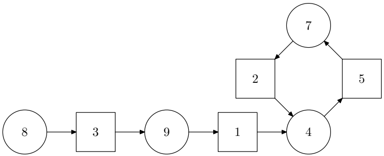
Positional Play

$$\pi_{\sigma,\tau,v} = \underbrace{\boxed{8}, 3, \boxed{9}, 1, (4, 5,}\ \boxed{7}, 2)^{\omega}$$

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

$$\pi_{\sigma,\tau,v} \mapsto \mathrm{val}_{\sigma,\tau}(v)$$



Positional Play

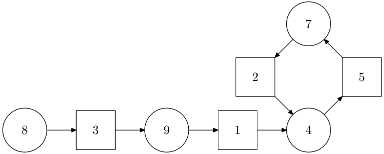
$$\pi_{\sigma,\tau,v} = \underbrace{8, 3, 9, 1, (4, 5,} 7, 2)^\omega$$

$$\mathrm{val}_{\sigma,\tau}(v) = ($$

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

$$\pi_{\sigma,\tau,v} \mapsto \mathrm{val}_{\sigma,\tau}(v)$$



Positional Play

$$\pi_{\sigma,\tau,v} = \underbrace{\boxed{8}, 3, \boxed{9}, 1, (4, 5,}\ \boxed{7}, 2)^\omega$$

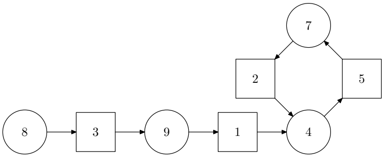
$$\mathrm{val}_{\sigma,\tau}(v) = (\boxed{7},$$

- cycle component

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

$$\pi_{\sigma,\tau,v} \mapsto \mathrm{val}_{\sigma,\tau}(v)$$



Positional Play

$$\pi_{\sigma,\tau,v} = \underbrace{\boxed{8}, 3, \boxed{9}, 1, (4, 5,}\ \boxed{7}, 2)^\omega$$

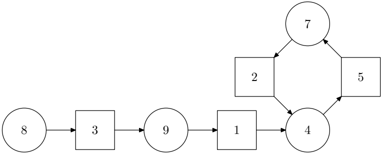
$$\mathrm{val}_{\sigma,\tau}(v) = (\boxed{7}, \boxed{\{8, 9\}},$$

- cycle component
- path component

Positional Plays and Valuations

Let σ player 0 strategy, τ player 1 strategy, v node.

$$\pi_{\sigma,\tau,v} \mapsto \mathrm{val}_{\sigma,\tau}(v)$$



Positional Play

$$\pi_{\sigma,\tau,v} = \underbrace{\boxed{8}, 3, \boxed{9}, 1, (4, 5,}\ \boxed{7}, 2)^{\omega}$$

$$\mathrm{val}_{\sigma,\tau}(v) = (\boxed{7}, \boxed{\{8, 9\}}, \boxed{6})$$

- cycle component
- path component
- length component

Comparing Valuations

$$(p_1, P_1, l_1) \prec (p_2, P_2, l_2)$$

iff

Comparing Valuations

$$(p_1, P_1, l_1) \prec (p_2, P_2, l_2)$$

iff

1. $(-1)^{p_1} p_1 < (-1)^{p_2} p_2$, i.e. $\ldots < 5 < 3 < 1 < 0 < 2 < 4 < \ldots$

   e.g. $(3, -, -) \prec (2, -, -)$

Comparing Valuations

$$(p_1, P_1, l_1) \prec (p_2, P_2, l_2)$$

iff

1. $(-1)^{p_1} p_1 < (-1)^{p_2} p_2$, i.e. $\ldots < 5 < 3 < 1 < 0 < 2 < 4 < \ldots$
   e.g. $(3, -, -) \prec (2, -, -)$
2. $p_1 = p_2$, and largest $p \in P_1 \triangle P_2$ is even & $p \in P_2$, or is odd & $p \in P_1$
   e.g. $(3, \{9, 8, 7, 6\}, -) \prec (3, \{9, 8, 6, 5\}, -)$ since $\{9, 8, 7, 6\} \triangle \{9, 8, 6, 5\} = \{7, 5\}$

Comparing Valuations

$$(p_1, P_1, l_1) \prec (p_2, P_2, l_2)$$

iff

1. $(-1)^{p_1} p_1 < (-1)^{p_2} p_2$, i.e. $\ldots < 5 < 3 < 1 < 0 < 2 < 4 < \ldots$

   e.g. $(3, -, -) \prec (2, -, -)$

2. $p_1 = p_2$, and largest $p \in P_1 \triangle P_2$ is even & $p \in P_2$, or is odd & $p \in P_1$

   e.g. $(3, \{9, 8, 7, 6\}, -) \prec (3, \{9, 8, 6, 5\}, -)$ since $\{9, 8, 7, 6\} \triangle \{9, 8, 6, 5\} = \{7, 5\}$

3. $p_1 = p_2$ and $P_1 = P_2$, and $(-1)^{p_1} l_1 > (-1)^{p_1} l_2$

   e.g. $(3, \{9, 8, 7, 6\}, 20) \prec (3, \{9, 8, 7, 6\}, 21)$ and $(2, \{9, 8, 7, 6\}, 21) \prec (2, \{9, 8, 7, 6\}, 20)$

Valuation of σ

Single player case:

$$\mathrm{val}_{\sigma}(v) = \mathrm{val}_{\sigma,\tau}(v), \qquad \tau \text{ trivial}$$

Valuation of σ

Single player case:

$$\mathrm{val}_{\sigma}(v) = \mathrm{val}_{\sigma,\tau}(v), \qquad \tau \text{ trivial}$$

Two player case:

$$\mathrm{val}_{\sigma}(v) = \min_{\tau} \mathrm{val}_{\sigma,\tau}(v)$$

Valuation of σ

Single player case:

$$\mathrm{val}_{\sigma}(v) = \mathrm{val}_{\sigma,\tau}(v), \qquad \tau \text{ trivial}$$

Two player case:

$$\mathrm{val}_{\sigma}(v) = \min_{\tau} \mathrm{val}_{\sigma,\tau}(v)$$

Facts

- val_{σ} polytime computable
- there is a single optimal counterstrategy τ_{σ} s.t. $\mathrm{val}_{\sigma}(v) = \mathrm{val}_{\sigma,\tau_{\sigma}}(v)$

Valuation of σ

Single player case:

$$\mathrm{val}_{\sigma}(v) = \mathrm{val}_{\sigma,\tau}(v), \qquad \tau \text{ trivial}$$

Two player case:

$$\mathrm{val}_{\sigma}(v) = \min_{\tau} \mathrm{val}_{\sigma,\tau}(v)$$

Facts

- val_{σ} polytime computable
- there is a single optimal counterstrategy τ_{σ} s.t. $\mathrm{val}_{\sigma}(v) = \mathrm{val}_{\sigma,\tau_{\sigma}}(v)$

Pre-ordering $\trianglelefteq$ on strategy valuations by point-wise comparison:

$$\sigma \trianglelefteq \sigma' \quad \Longleftrightarrow \quad \forall v.\ \mathrm{val}_{\sigma}(v) \preceq \mathrm{val}_{\sigma'}(v)$$

Scheme (recall)

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies ✓
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$

with two properties...

1. $\trianglelefteq$-optimal strategy σ induces winning sets and winning strategies
2. σ not $\trianglelefteq$-optimal implies $\sigma \lhd \textsc{Improve}(\sigma)$

Strategy Improvement

1: **while** σ is not optimal **do**
2: $\quad\sigma \leftarrow \textsc{Improve}(\sigma)$
3: **end while**

(Improving) Switches

Let $E_0 = E \cap (V_0 \times V)$ set of player 0 edges.
A σ-switch is an edge $e \in E_0 \setminus \sigma$ not chosen by σ.

(Improving) Switches

Let $E_0 = E \cap (V_0 \times V)$ set of player 0 edges.
A σ-switch is an edge $e \in E_0 \setminus \sigma$ not chosen by σ.

Facts about switches

- Comparability: $\mathrm{val}_\sigma \trianglelefteq \mathrm{val}_{\sigma[e]}$ or $\mathrm{val}_{\sigma[e]} \trianglelefteq \mathrm{val}_\sigma$ for every σ-switch e.
- Easy Check: $\mathrm{val}_\sigma \lhd \mathrm{val}_{\sigma[(v,w)]}$ iff $\mathrm{val}_\sigma(\sigma(v)) \prec \mathrm{val}_\sigma(w)$.

Improving Switches: $I(\sigma) = \{e \mid \mathrm{val}_\sigma \lhd \mathrm{val}_{\sigma[e]}\}$

(Improving) Switches

Let $E_0 = E \cap (V_0 \times V)$ set of player 0 edges.
A σ-switch is an edge $e \in E_0 \setminus \sigma$ not chosen by σ.

Facts about switches

- Comparability: $\mathrm{val}_\sigma \trianglelefteq \mathrm{val}_{\sigma[e]}$ or $\mathrm{val}_{\sigma[e]} \trianglelefteq \mathrm{val}_\sigma$ for every σ-switch e.
- Easy Check: $\mathrm{val}_\sigma \lhd \mathrm{val}_{\sigma[(v,w)]}$ iff $\mathrm{val}_\sigma(\sigma(v)) \prec \mathrm{val}_\sigma(w)$.

Improving Switches: $I(\sigma) = \{e \mid \mathrm{val}_\sigma \lhd \mathrm{val}_{\sigma[e]}\}$

Theorem

- Switching: $\emptyset \subsetneq J \subseteq I(\sigma)$ implies $\mathrm{val}_\sigma \lhd \mathrm{val}_{\sigma[J]}$.
- Optimality: $I(\sigma) = \emptyset$ implies σ is optimal.

(Improving) Switches

Let $E_0 = E \cap (V_0 \times V)$ set of player 0 edges.
A σ-switch is an edge $e \in E_0 \setminus \sigma$ not chosen by σ.

Facts about switches

- Comparability: $\mathrm{val}_\sigma \trianglelefteq \mathrm{val}_{\sigma[e]}$ or $\mathrm{val}_{\sigma[e]} \trianglelefteq \mathrm{val}_\sigma$ for every σ-switch e.
- Easy Check: $\mathrm{val}_\sigma \triangleleft \mathrm{val}_{\sigma[(v,w)]}$ iff $\mathrm{val}_\sigma(\sigma(v)) \prec \mathrm{val}_\sigma(w)$.

Improving Switches: $I(\sigma) = \{e \mid \mathrm{val}_\sigma \triangleleft \mathrm{val}_{\sigma[e]}\}$

Theorem

- Switching: $\emptyset \subsetneq J \subseteq I(\sigma)$ implies $\mathrm{val}_\sigma \triangleleft \mathrm{val}_{\sigma[J]}$.
- Optimality: $I(\sigma) = \emptyset$ implies σ is optimal.

$\textsc{Improve}(\sigma) = \sigma[J]$ for some $\emptyset \subsetneq J \subseteq I(\sigma)$

Scheme (recall)

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies ✓
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$ ✓

with two properties...

1. $\trianglelefteq$-optimal strategy σ induces winning sets and winning strategies
2. σ not $\trianglelefteq$-optimal implies $\sigma \triangleleft \textsc{Improve}(\sigma)$ ✓

Strategy Improvement

```
1: while σ is not optimal do
2:     σ ← IMPROVE(σ)
3: end while
```

Winning Sets and Strategies

Theorem

Let σ be an optimal strategy.

1. $W_0 = \{v \mid \mathrm{val}_\sigma(v) = (w, \_, \_) \textit{ and } w \textit{ even}\}$
2. $W_1 = \{v \mid \mathrm{val}_\sigma(v) = (w, \_, \_) \textit{ and } w \textit{ odd}\}$
3. σ *is a winning strategy for player 0 on* W_0
4. τ_σ *is a winning strategy for player 1 on* W_1

Scheme (recall)

Two ingredients...

1. Pre-ordering $\trianglelefteq$ on positional player 0 strategies ✓
2. Improvement operator $\textsc{Improve} : \mathcal{S}_0 \to \mathcal{S}_0$ ✓

with two properties...

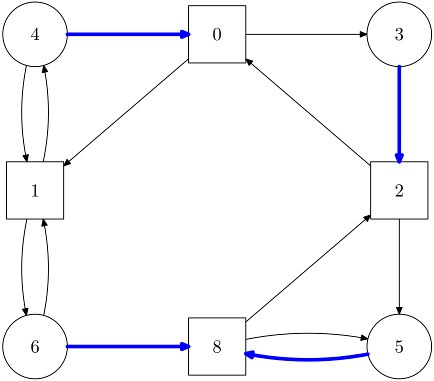
1. $\trianglelefteq$-optimal strategy σ induces winning sets and strategies ✓
2. σ not $\trianglelefteq$-optimal implies $\sigma \lhd \textsc{Improve}(\sigma)$ ✓

Strategy Improvement

1: **while** σ is not optimal **do**
2: $\quad\sigma \leftarrow \textsc{Improve}(\sigma)$
3: **end while**

Example

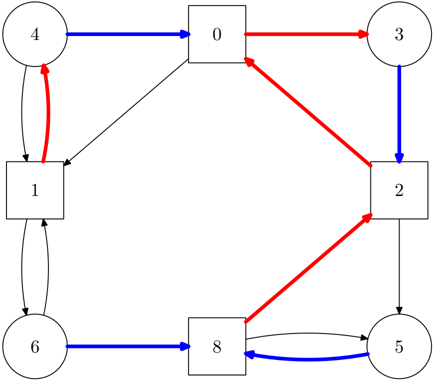
Initial strategy



- $\mathrm{val}_\sigma(6) =$
- $\mathrm{val}_\sigma(5) =$

Example

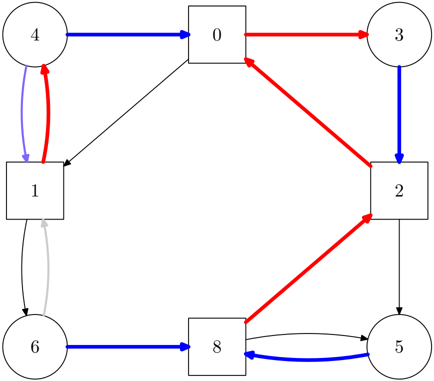
Counter strategy / Valuation



- $\mathrm{val}_\sigma(6) = (3, \{8, 6\}, 4)$ induced by $\pi = 6, 8, 2, 0, (3, 2, 0)^\omega$
- $\mathrm{val}_\sigma(5) = (3, \{8, 5\}, 4)$ induced by $\pi = 5, 8, 2, 0, (3, 2, 0)^\omega$

Example

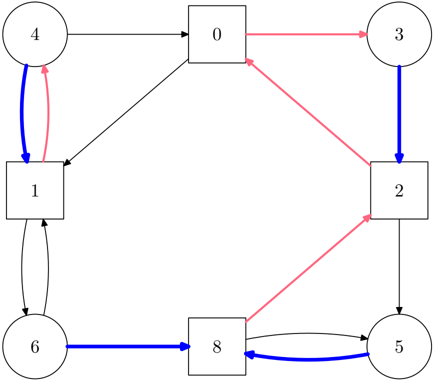
Improving switches



- $\mathrm{val}_\sigma(6) = (\boxed{3}, \{\boxed{8}, \boxed{6}\}, \boxed{4})$ induced by $\pi = \boxed{6}, \boxed{8}, \boxed{2}, \boxed{0}, (\boxed{3}, 2, 0)^\omega$
- $\mathrm{val}_\sigma(5) = (\boxed{3}, \{\boxed{8}, \boxed{5}\}, \boxed{4})$ induced by $\pi = \boxed{5}, \boxed{8}, \boxed{2}, \boxed{0}, (\boxed{3}, 2, 0)^\omega$

Example

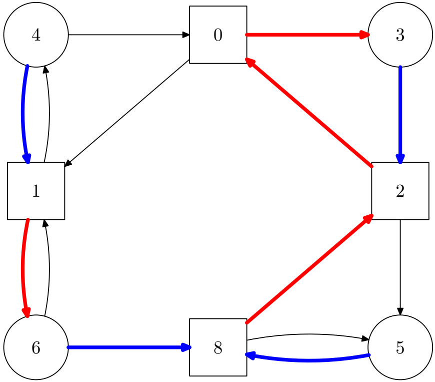
Next strategy



- $\mathrm{val}_\sigma(6) =$
- $\mathrm{val}_\sigma(5) =$

Example

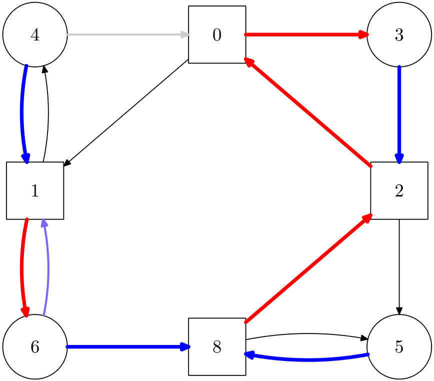
Next counter strategy / Next valuation



- $\mathrm{val}_\sigma(6) = (\boxed{3}, \{\boxed{8}, \boxed{6}\}, \boxed{4})$ induced by $\pi = \boxed{6}, \boxed{8}, \boxed{2}, \boxed{0}, (\boxed{3}, 2, 0)^\omega$
- $\mathrm{val}_\sigma(5) = (\boxed{3}, \{\boxed{8}, \boxed{5}\}, \boxed{4})$ induced by $\pi = \boxed{5}, \boxed{8}, \boxed{2}, \boxed{0}, (\boxed{3}, 2, 0)^\omega$

Example

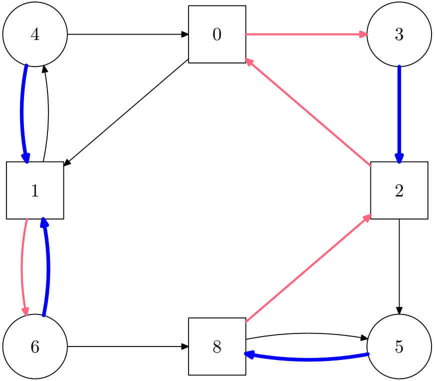
Improving switches



- $\mathrm{val}_\sigma(6) = (\boxed{3}, \{\boxed{8}, \boxed{6}\}, \boxed{4})$ induced by $\pi = \boxed{6}, \boxed{8}, \boxed{2}, \boxed{0}, (\boxed{3}, 2, 0)^\omega$
- $\mathrm{val}_\sigma(5) = (\boxed{3}, \{\boxed{8}, \boxed{5}\}, \boxed{4})$ induced by $\pi = \boxed{5}, \boxed{8}, \boxed{2}, \boxed{0}, (\boxed{3}, 2, 0)^\omega$

Example

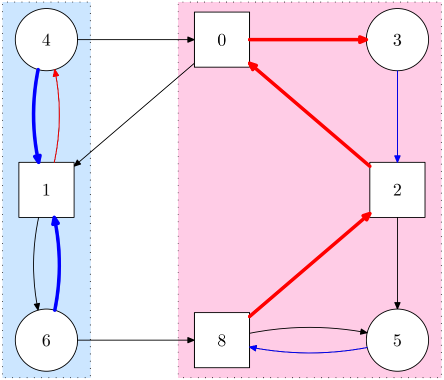
Final strategy



- $\mathrm{val}_\sigma(6) =$
- $\mathrm{val}_\sigma(5) =$

Example

Final counter strategy / Final valuation



- $\mathrm{val}_\sigma(6) = (4, \{6\}, 2)$ induced by $\pi = 6, 1, (4, 1)^\omega$
- $\mathrm{val}_\sigma(5) = (3, \{8, 5\}, 4)$ induced by $\pi = 5, 8, 2, 0, (3, 2, 0)^\omega$

Complexity

Assume that

- checking for $\trianglelefteq$-optimality, and
- computing the improvement operator Improve

both require polynomial time.

Complexity of Strategy Iteration

Complexity

Assume that

- checking for $\trianglelefteq$-optimality, and
- computing the improvement operator Improve

both require polynomial time.

Complexity of Strategy Iteration essentially depends on the number of iterations!

Improvement Rules

Strategy Iteration is parameterized by an improvement rule.

$$\textsc{Improve}(\sigma) = \sigma[J] \text{ for some } \emptyset \subsetneq J \subseteq I(\sigma)$$

Improvement Rule = method of chosing improving switches

- Single-Switching vs. Multi-Switching
- Deterministic vs. Randomized
- Memorizing vs. Oblivious

Diameter

Question: theoretically possible to have polynomially many iterations?

Let G be a game and n be the number of nodes.

Definition: the diameter of G is the least number of iterations required to solve G

Diameter Theorem

The diameter of G is less or equal to n.

Switch All

Standard improvement rule = simultaneous best local improvement

$$\textsc{Switch-All}(\sigma) : v \mapsto \operatorname*{argmax}_{w \in vE} \mathrm{val}_{\sigma}(w)$$

Switch All

Standard improvement rule = simultaneous best local improvement

$$\textsc{Switch-All}(\sigma) : v \mapsto \operatorname*{argmax}_{w \in vE} \mathrm{val}_\sigma(w)$$

Theorem

One player PGs can be solved in $\mathcal{O}(n)$ iterations by $\textsc{Switch-All}$ strategy improvement.

Lower Bounds

Sink Parity Games

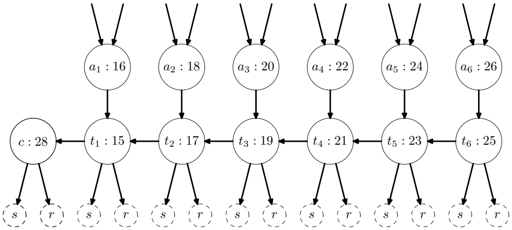
Definition: PG is sink PG iff

1. only one cycle component appears in strategy iteration
2. cycle component has least priority in the game, and is odd

Consequences:

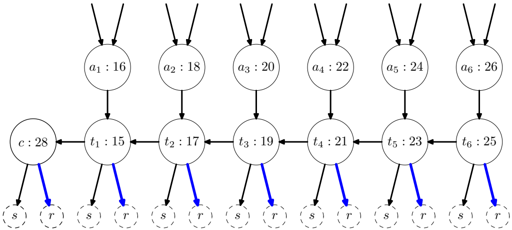
- Game is completely won by player 1
- Cycle component and path length component irrelevant
- Strategy Iteration = optimization of paths leading to the sink

Deceleration Lane



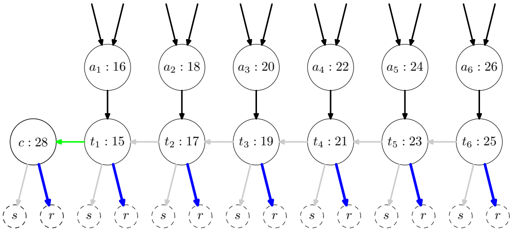
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point:

Deceleration Lane



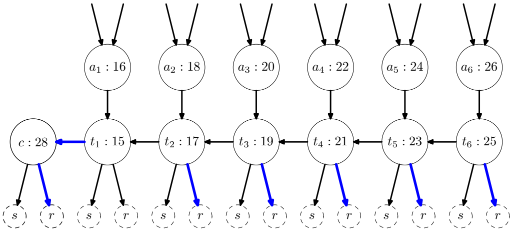
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_6

Deceleration Lane



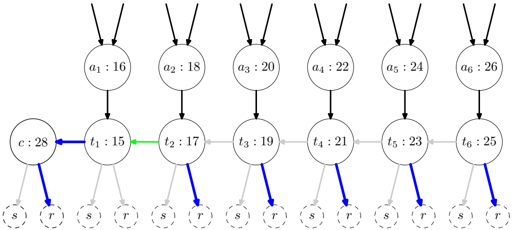
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_6

Deceleration Lane



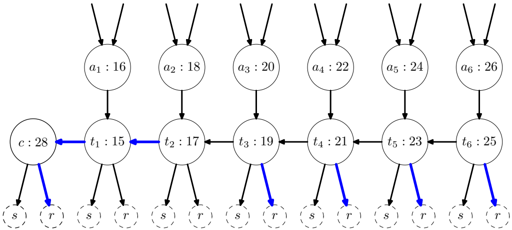
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_1

Deceleration Lane



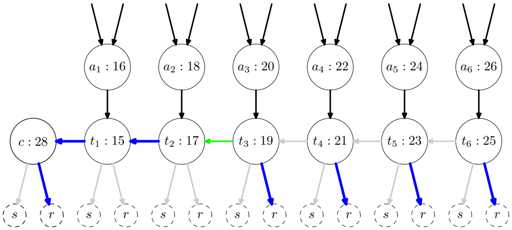
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_1

Deceleration Lane



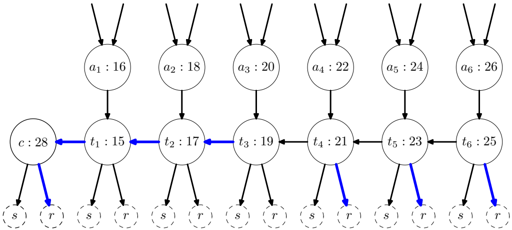
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_2

Deceleration Lane



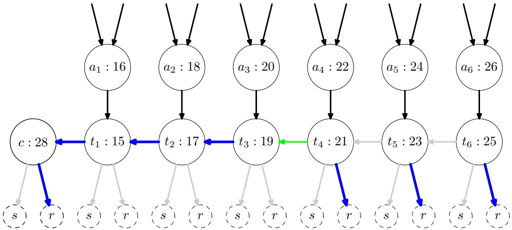
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_2

Deceleration Lane



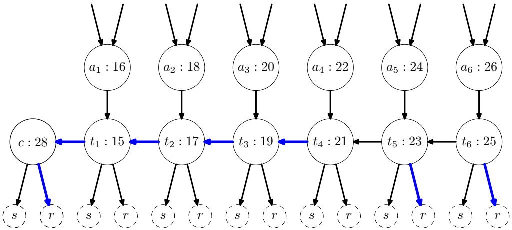
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_3

Deceleration Lane



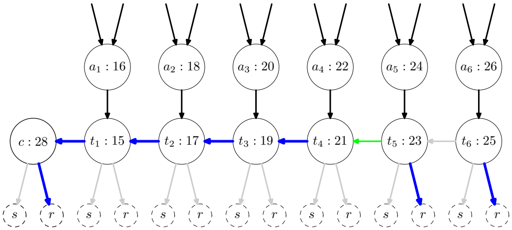
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_3

Deceleration Lane



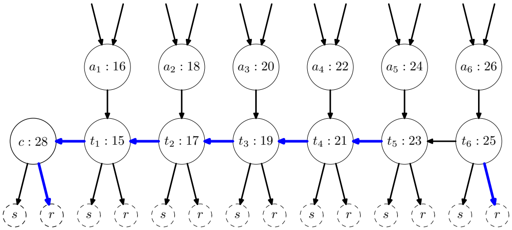
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_4

Deceleration Lane



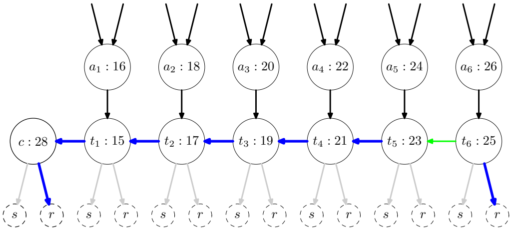
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_4

Deceleration Lane



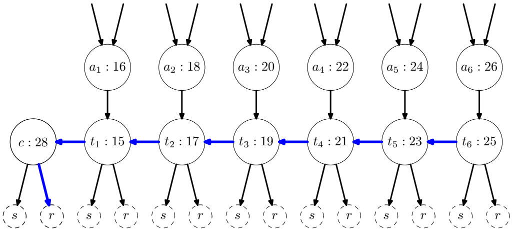
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_5

Deceleration Lane



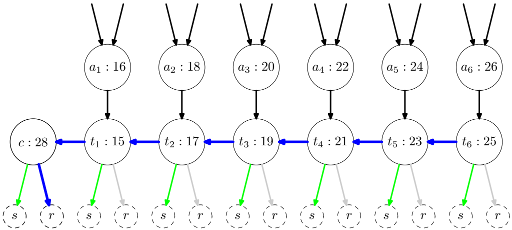
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_5

Deceleration Lane



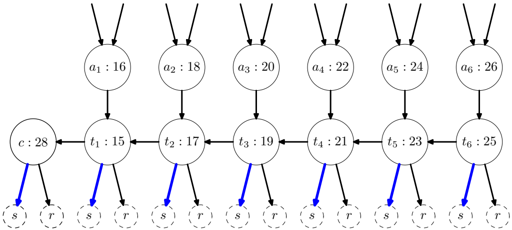
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_6

Deceleration Lane



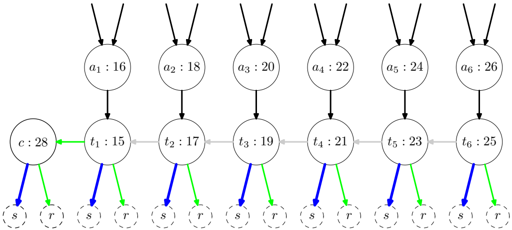
- Assume $\mathrm{val}_\sigma(r) \prec \mathrm{val}_\sigma(s)$
- Best entry point: a_6

Deceleration Lane



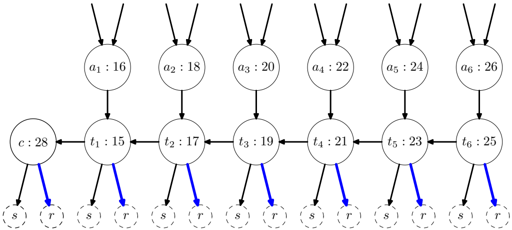
- Assume $\mathrm{val}_\sigma(r) \prec \mathrm{val}_\sigma(s)$
- Best entry point: a_6

Deceleration Lane



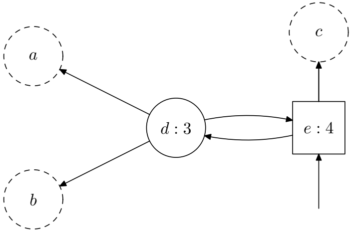
- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_6

Deceleration Lane



- Assume $\mathrm{val}_\sigma(s) \prec \mathrm{val}_\sigma(r)$
- Best entry point: a_6

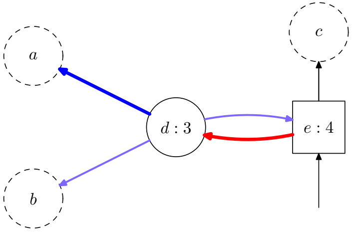
Cycle Gadget



Situation: Player 1 controlled, player 0 dominated cycle

- Ordering:
- $\mathrm{val}_\sigma(e) =$
- $\mathrm{val}_\sigma(d) =$

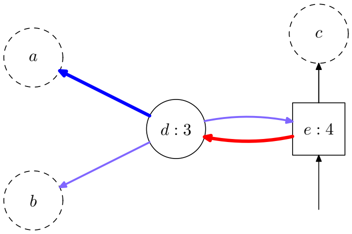
Cycle Gadget



Player 0 moves out, player 1 moves in ("cycle open")

- Ordering: $\mathrm{val}_\sigma(a) \prec \mathrm{val}_\sigma(e) \prec \mathrm{val}_\sigma(b) \lll \mathrm{val}_\sigma(c)$
- $\mathrm{val}_\sigma(e) = \mathrm{val}_\sigma(d) \cup \{e\} = \mathrm{val}_\sigma(a) \cup \{d, e\}$
- $\mathrm{val}_\sigma(d) = \mathrm{val}_\sigma(a) \cup \{d\}$

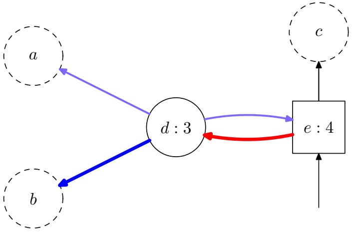
Cycle Gadget



Node c has the highest valuation, however, best local update is b

- Ordering: $\mathrm{val}_\sigma(a) \prec \mathrm{val}_\sigma(e) \prec \mathrm{val}_\sigma(b) \lll \mathrm{val}_\sigma(c)$
- $\mathrm{val}_\sigma(e) = \mathrm{val}_\sigma(d) \cup \{e\} = \mathrm{val}_\sigma(a) \cup \{d, e\}$
- $\mathrm{val}_\sigma(d) = \mathrm{val}_\sigma(a) \cup \{d\}$

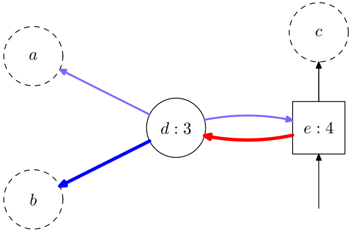
Cycle Gadget



Player 0 still moves out

- Ordering: $\text{val}_\sigma(b) \prec \text{val}_\sigma(e) \prec \text{val}_\sigma(a) \lll \text{val}_\sigma(c)$
- $\text{val}_\sigma(e) = \text{val}_\sigma(d) \cup \{e\} = \text{val}_\sigma(b) \cup \{d, e\}$
- $\text{val}_\sigma(d) = \text{val}_\sigma(b) \cup \{d\}$

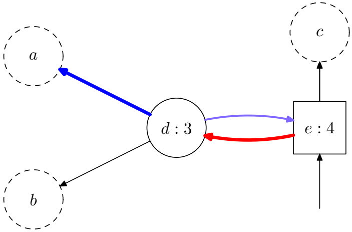
Cycle Gadget



Node c still has highest valuation, however, best local update is a

- Ordering: $\mathrm{val}_\sigma(b) \prec \mathrm{val}_\sigma(e) \prec \mathrm{val}_\sigma(a) \lll \mathrm{val}_\sigma(c)$
- $\mathrm{val}_\sigma(e) = \mathrm{val}_\sigma(d) \cup \{e\} = \mathrm{val}_\sigma(b) \cup \{d, e\}$
- $\mathrm{val}_\sigma(d) = \mathrm{val}_\sigma(b) \cup \{d\}$

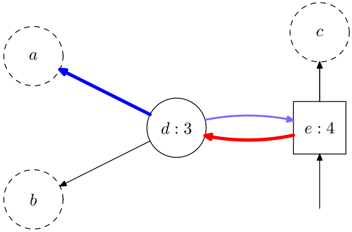
Cycle Gadget



Still moving out...

- Ordering: $\mathrm{val}_\sigma(b) \prec \mathrm{val}_\sigma(a) \prec \mathrm{val}_\sigma(e) \lll \mathrm{val}_\sigma(c)$
- $\mathrm{val}_\sigma(e) = \mathrm{val}_\sigma(d) \cup \{e\} = \mathrm{val}_\sigma(a) \cup \{d, e\}$
- $\mathrm{val}_\sigma(d) = \mathrm{val}_\sigma(a) \cup \{d\}$

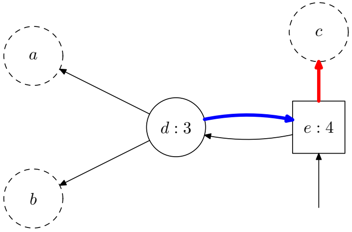
Cycle Gadget



Only improving edge is d ("closing the cycle")

- Ordering: $\mathrm{val}_\sigma(b) \prec \mathrm{val}_\sigma(a) \prec \mathrm{val}_\sigma(e) \lll \mathrm{val}_\sigma(c)$
- $\mathrm{val}_\sigma(e) = \mathrm{val}_\sigma(d) \cup \{e\} = \mathrm{val}_\sigma(a) \cup \{d, e\}$
- $\mathrm{val}_\sigma(d) = \mathrm{val}_\sigma(a) \cup \{d\}$

Cycle Gadget



Cycle closed, player 1 forced to move out

- Ordering: $\mathrm{val}_\sigma(b) \prec \mathrm{val}_\sigma(a) \lll \mathrm{val}_\sigma(c) \prec \mathrm{val}_\sigma(e)$
- $\mathrm{val}_\sigma(e) = \mathrm{val}_\sigma(c) \cup \{e\}$
- $\mathrm{val}_\sigma(d) = \mathrm{val}_\sigma(c) \cup \{d, e\}$

Binary Counting

101011

Binary Counting

101011

↓

Setting

↓

101111

Binary Counting

101011

↓

Setting

↓

101111

↓

Resetting

↓

101100

Binary Counting

101011

$\downarrow$

Setting

$\downarrow$

101111

$\downarrow$

Resetting

$\downarrow$

101100

$\downarrow$

Activating

$\downarrow$

101100

Binary Counting

101011
101011

101011
↓
Setting
↓
101111
↓
Resetting
↓
101100
↓
Activating
↓
101100

Binary Counting

101011
↓
Setting
↓
101111
↓
Resetting
↓
101100
↓
Activating
↓
101100

101011
101011
↓
Setting
↓
101111
101011

Binary Counting

101011
↓
Setting
↓
101111
↓
Resetting
↓
101100
↓
Activating
↓
101100

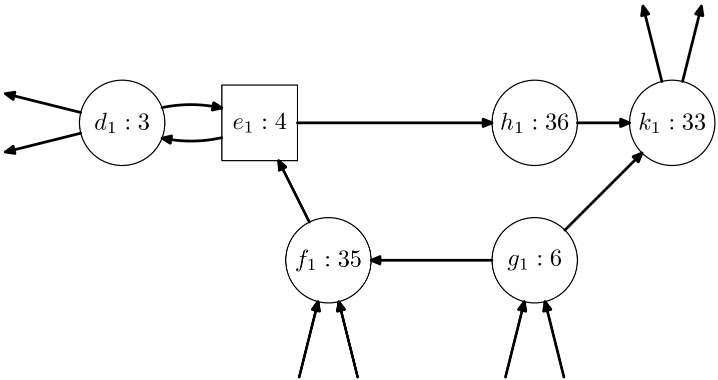
101011
101011
↓
Setting
↓
101111
101011
↓
Resetting
↓
101100
101000

Binary Counting

101011
↓
Setting
↓
101111
↓
Resetting
↓
101100
↓
Activating
↓
101100

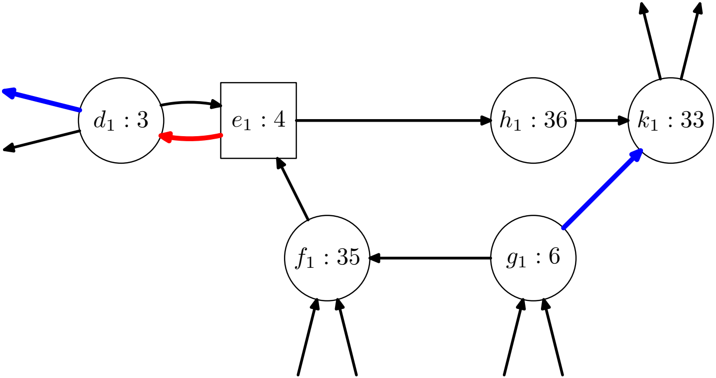
101011
101011
↓
Setting
↓
101111
101011
↓
Resetting
↓
101100
101000
↓
Activating
↓
101100
101100

Cycle Gate Gadget



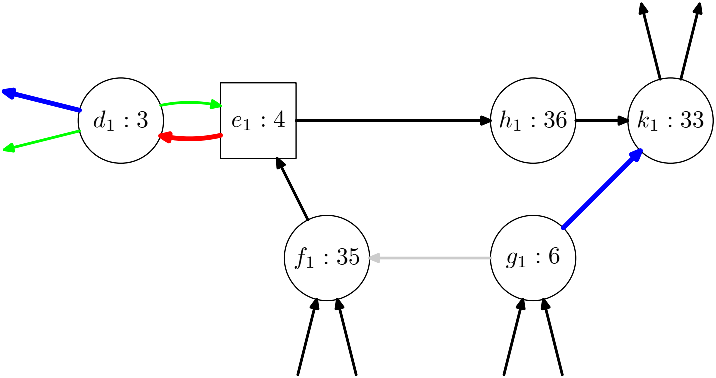
g_1 node serves as activation control

Cycle Gate Gadget



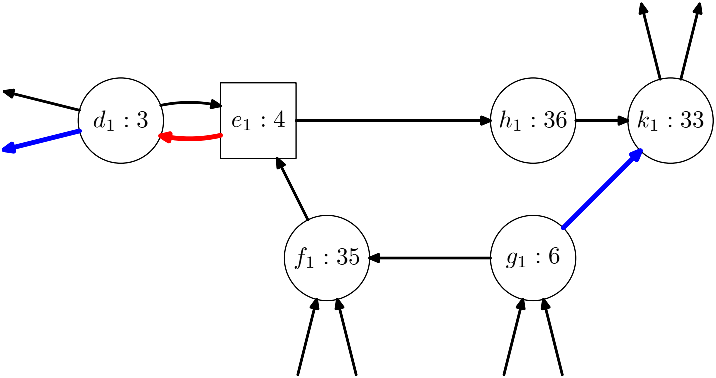
g_1 node serves as activation control

Cycle Gate Gadget



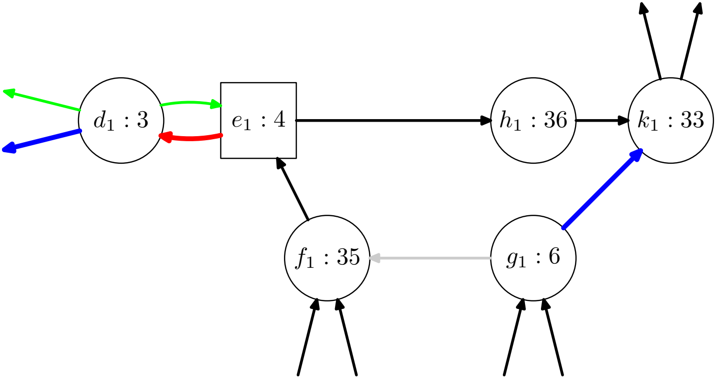
g_1 node serves as activation control

Cycle Gate Gadget



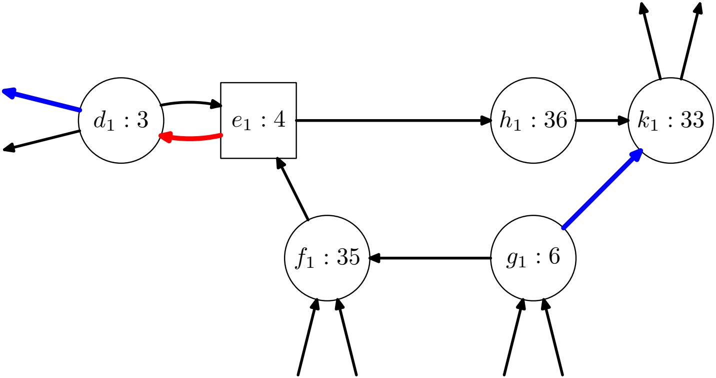
g_1 node serves as activation control

Cycle Gate Gadget



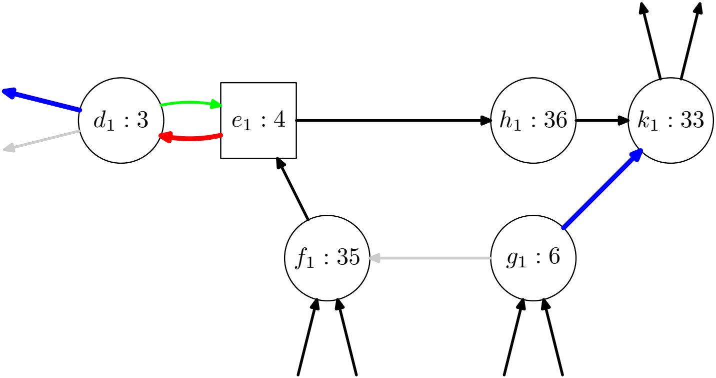
g_1 node serves as activation control

Cycle Gate Gadget



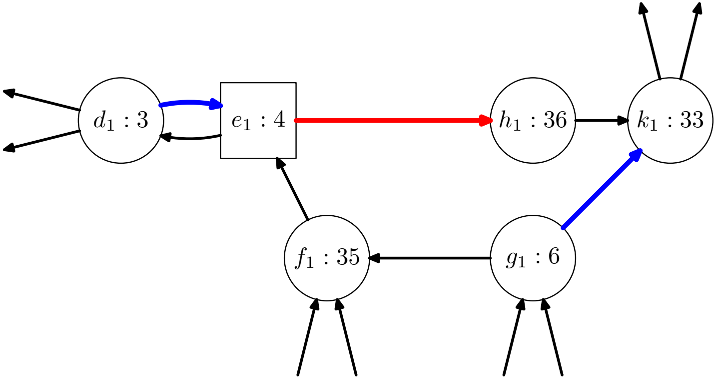
g_1 node serves as activation control

Cycle Gate Gadget



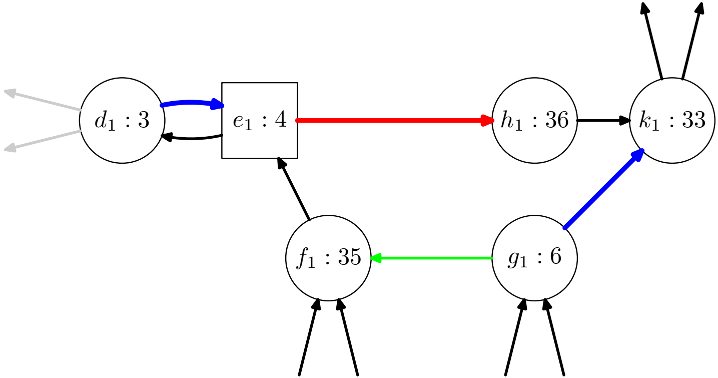
g_1 node serves as activation control

Cycle Gate Gadget



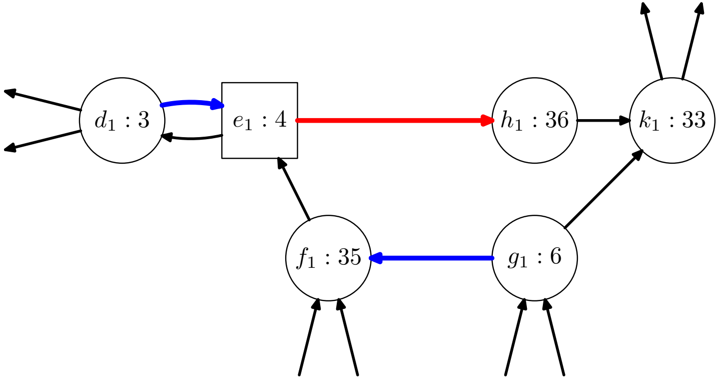
g_1 node serves as activation control

Cycle Gate Gadget



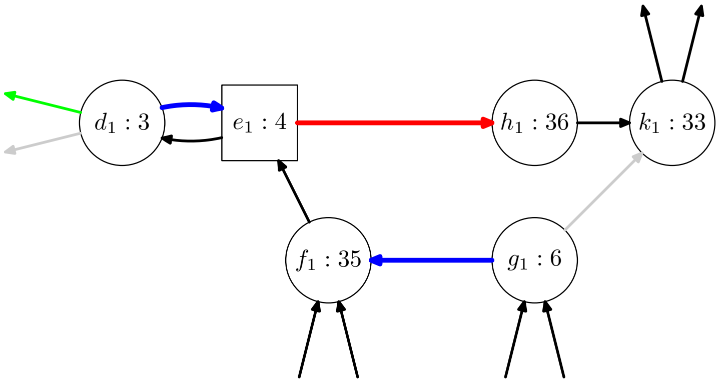
g_1 node serves as activation control

Cycle Gate Gadget



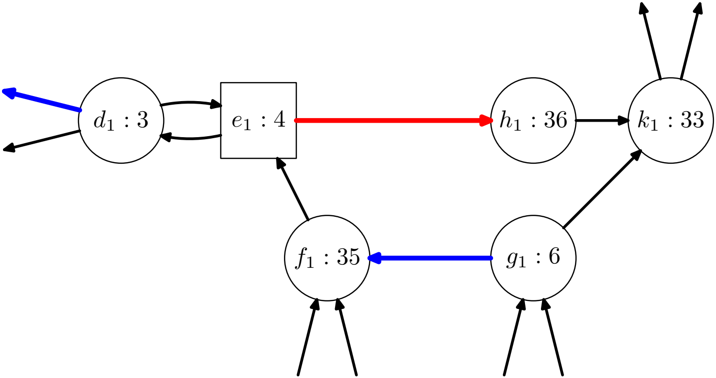
g_1 node serves as activation control

Cycle Gate Gadget



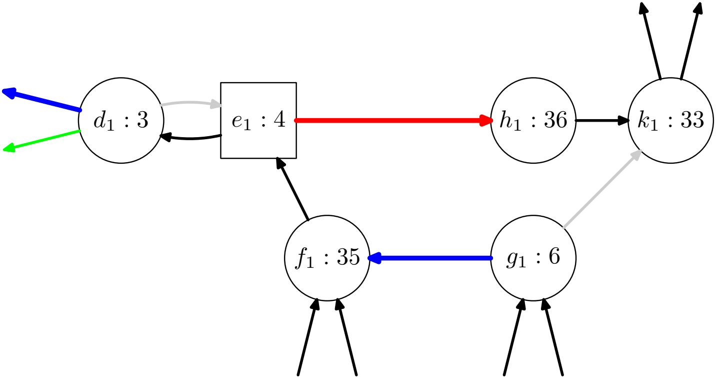
g_1 node serves as activation control

Cycle Gate Gadget



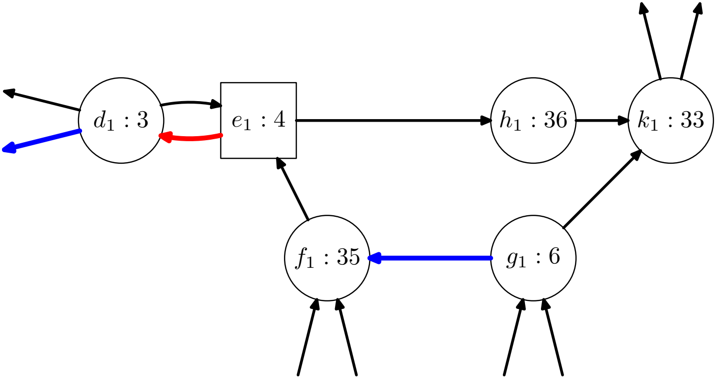
g_1 node serves as activation control

Cycle Gate Gadget



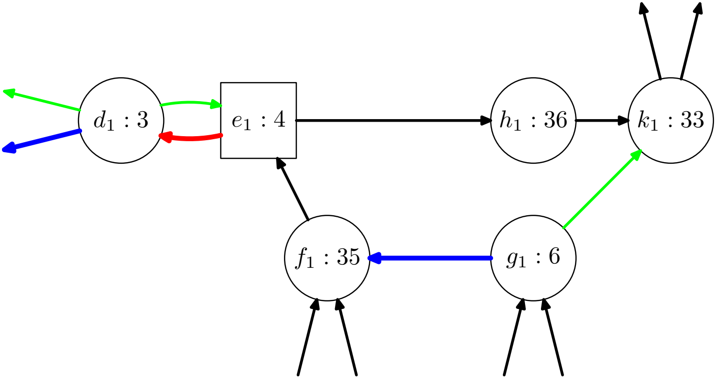
g_1 node serves as activation control

Cycle Gate Gadget



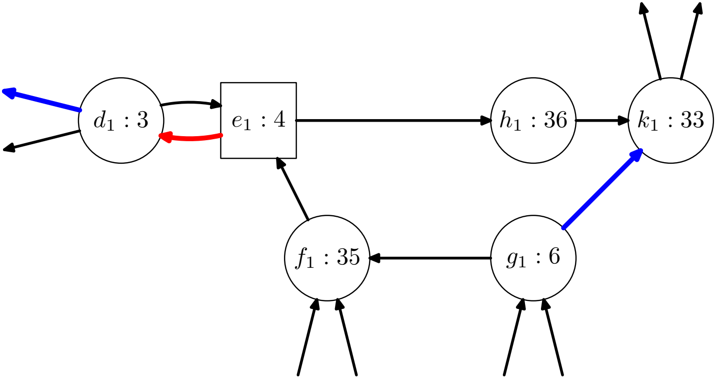
g_1 node serves as activation control

Cycle Gate Gadget



g_1 node serves as activation control

Cycle Gate Gadget

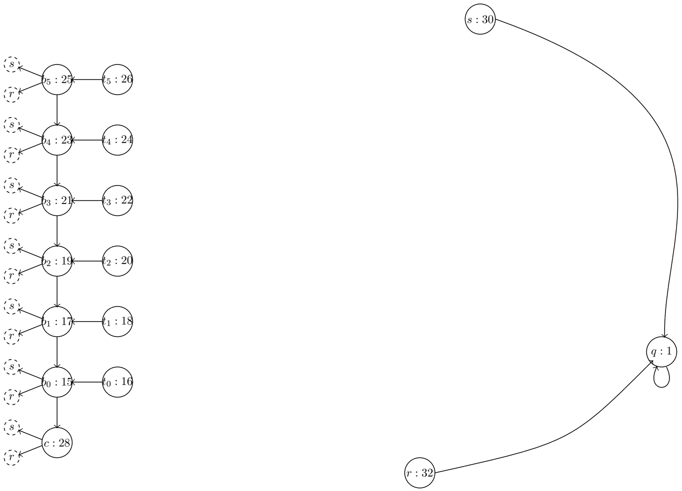


g_1 node serves as activation control

Full Construction

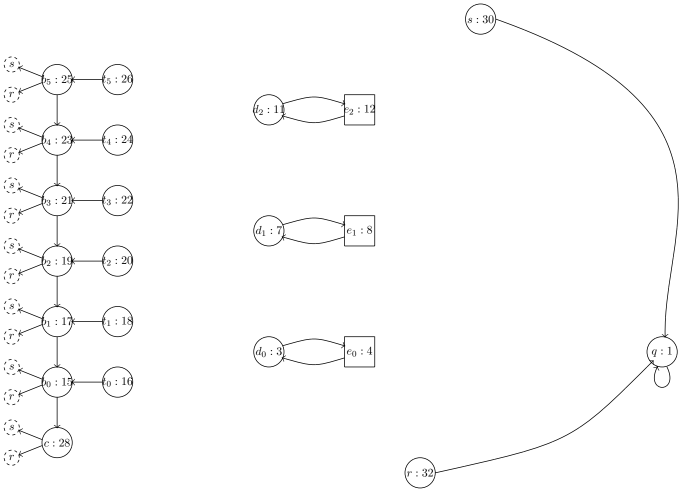
Whole graph consists of a simple cycle,

Full Construction



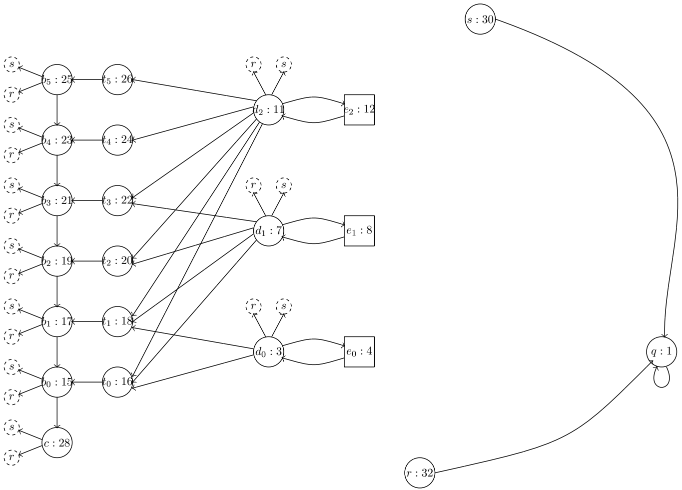
a deceleration lane,

Full Construction



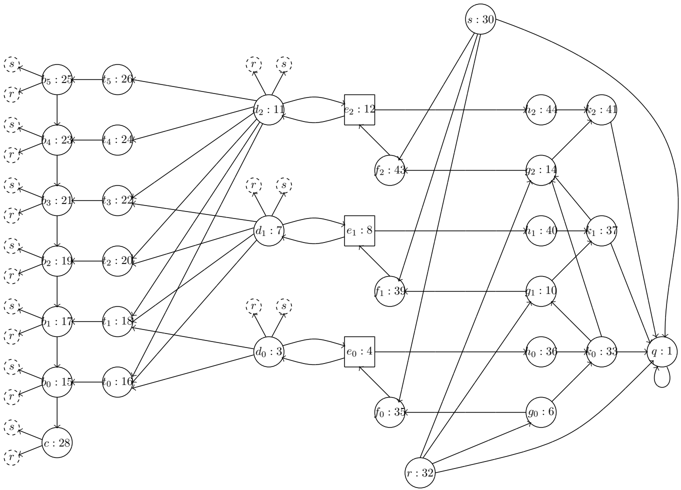
simple (bit-saving) cycles

Full Construction



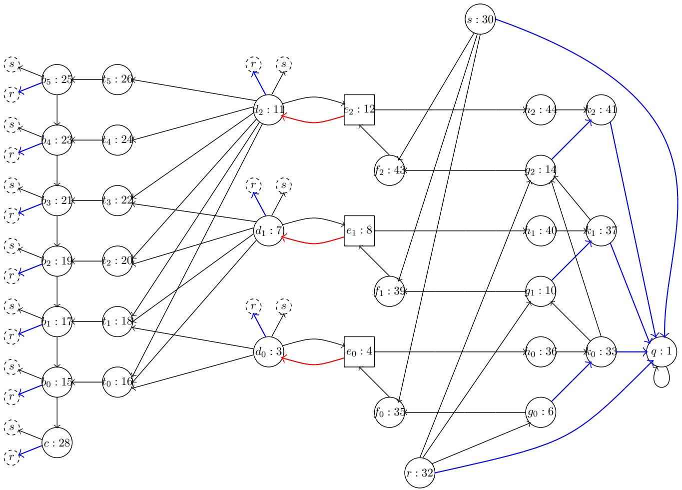
simple (bit-saving) cycles that are connected to the lane,

Full Construction



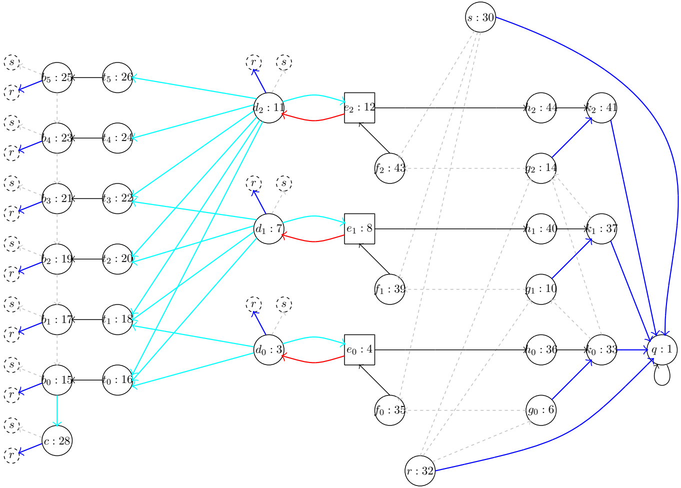
and cycle associated structures.

Full Construction



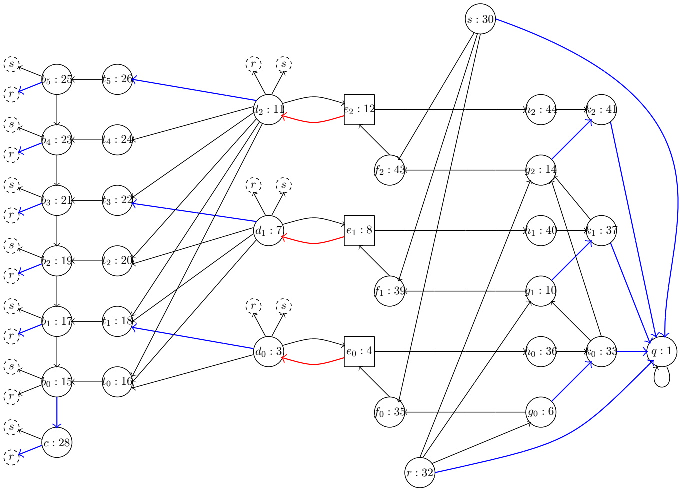
Initial Strategy, heuristic: Maximize local reward.

Full Construction



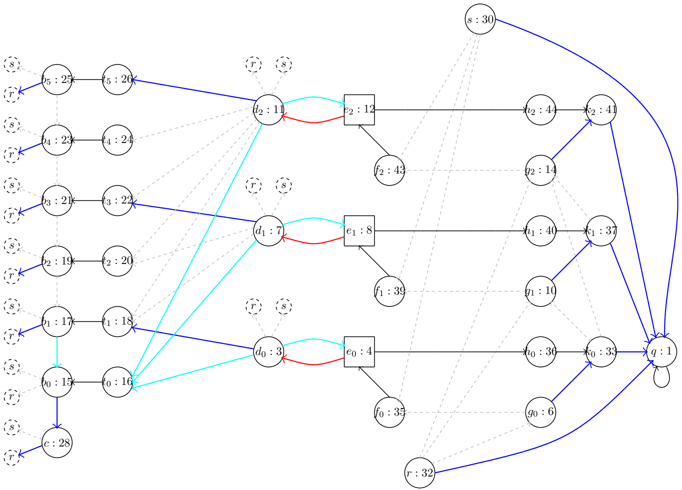
Lane improves iteratively, all cycles are occupied thereby.

Full Construction



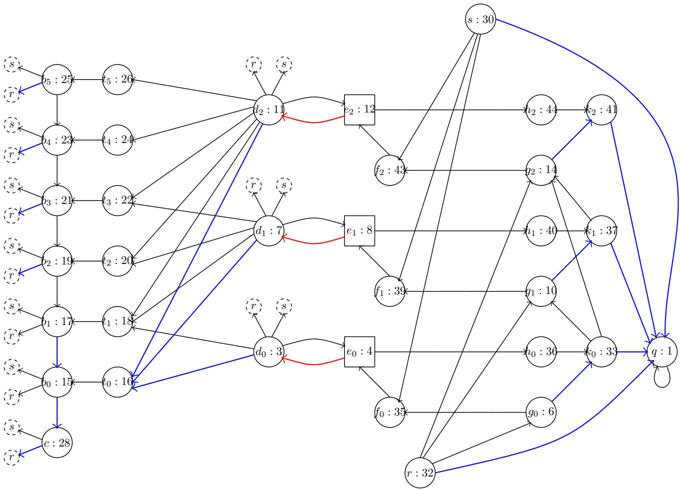
Lane improves iteratively, all cycles are occupied thereby.

Full Construction



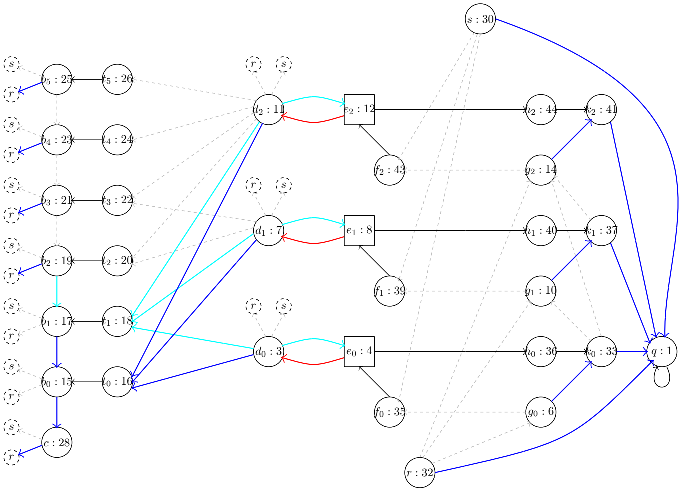
Lane improves iteratively, all cycles are occupied thereby.

Full Construction



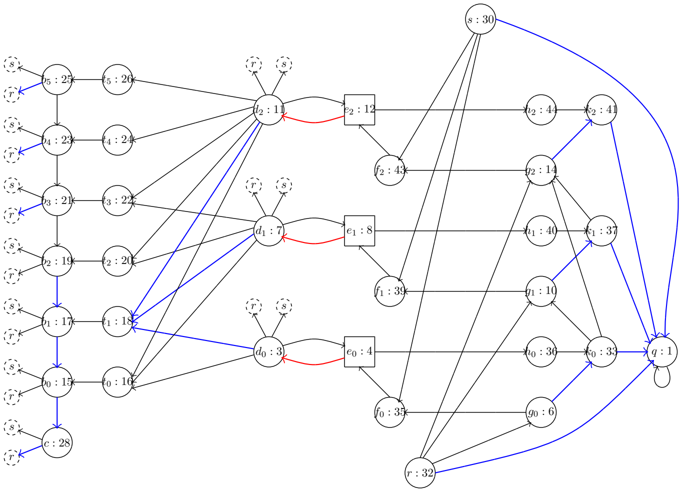
Lane improves iteratively, all cycles are occupied thereby.

Full Construction



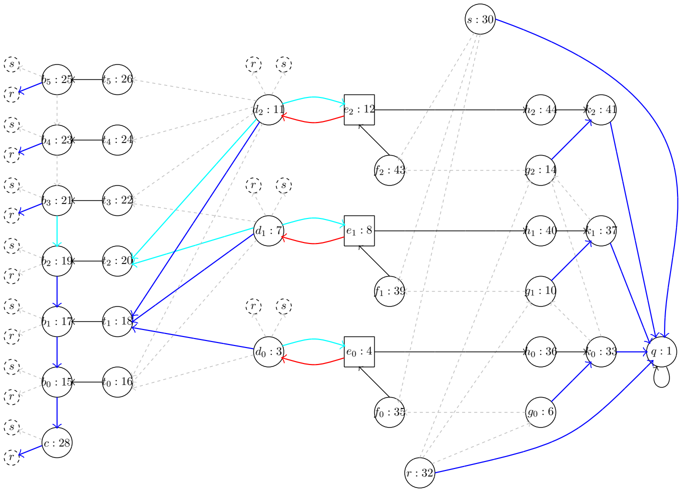
Lane improves iteratively, all cycles are occupied thereby.

Full Construction



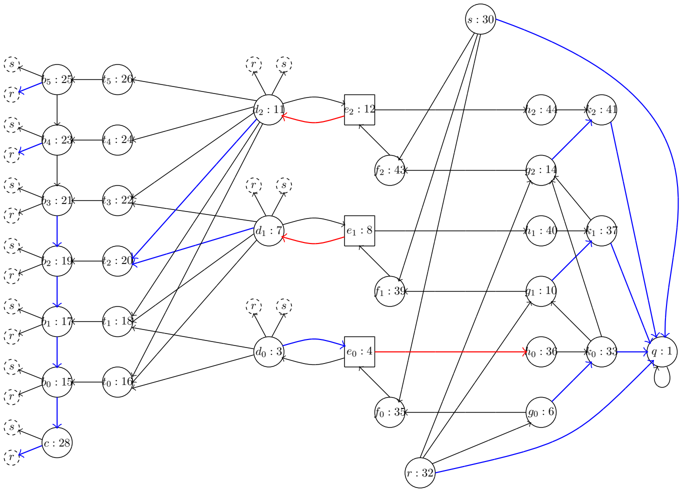
Lane improves iteratively, all cycles are occupied thereby.

Full Construction



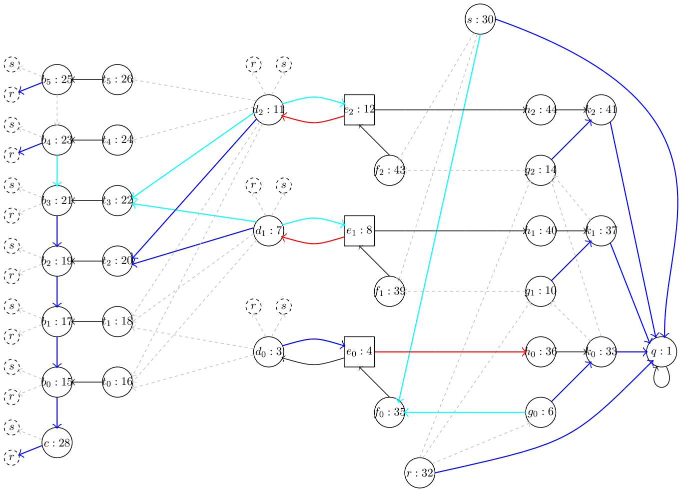
First cycle cannot improve furthermore to the lane.

Full Construction



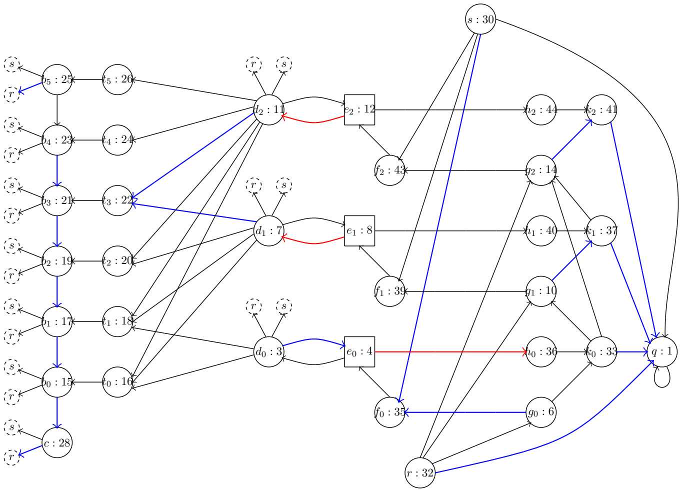
First cycles closes, forcing player 1 to leave it.

Full Construction



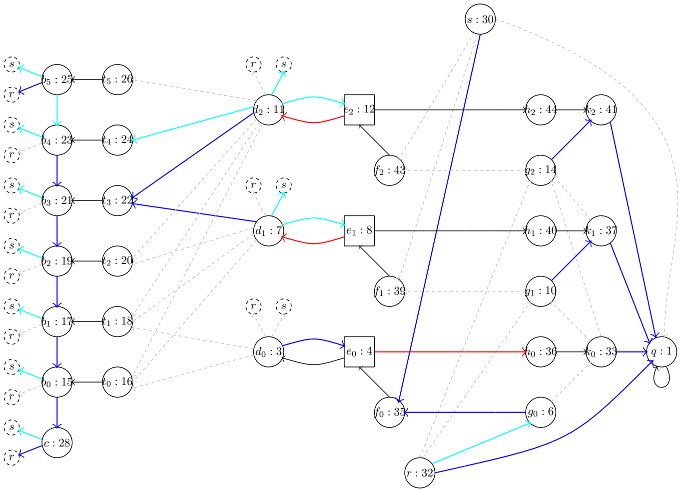
First cycles closes, forcing player 1 to leave it.

Full Construction



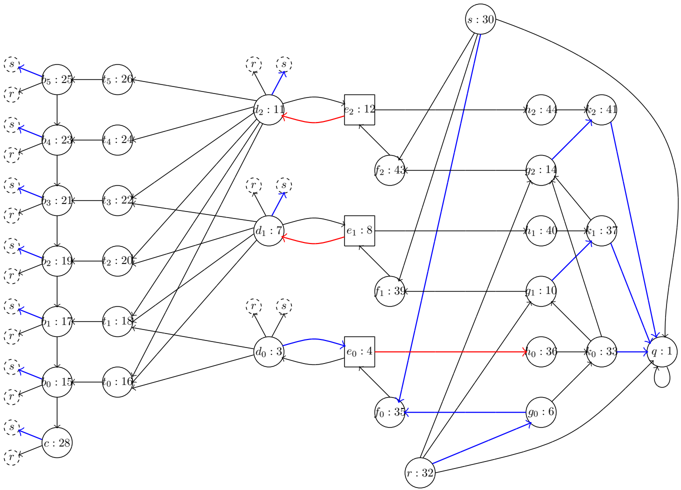
First cycles closes, forcing player 1 to leave it.

Full Construction



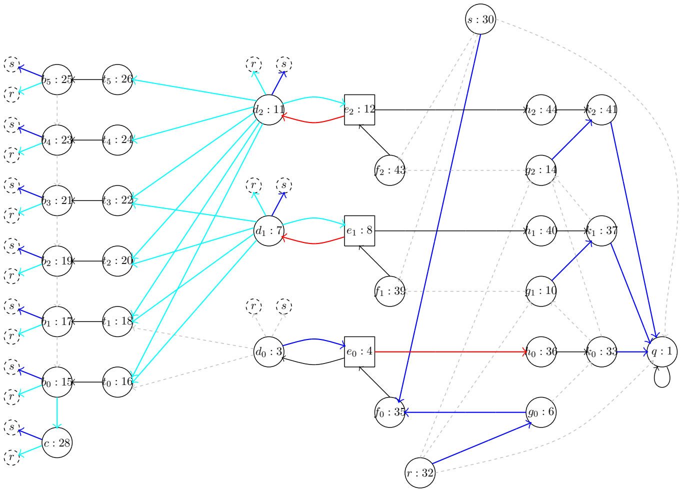
First cycles closes, forcing player 1 to leave it.

Full Construction



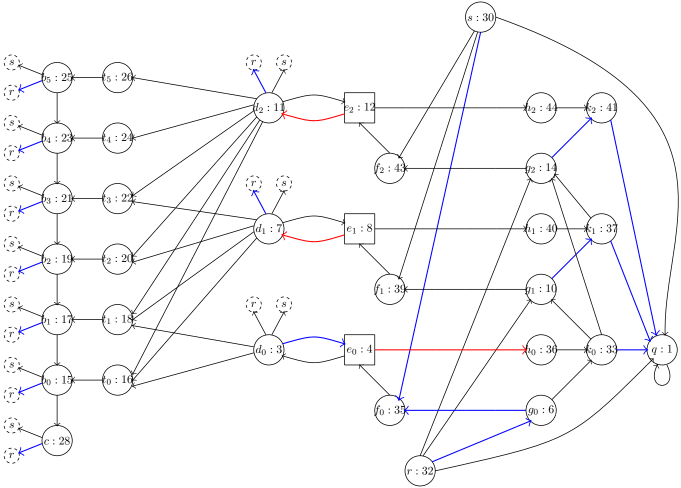
First cycles closes, forcing player 1 to leave it.

Full Construction



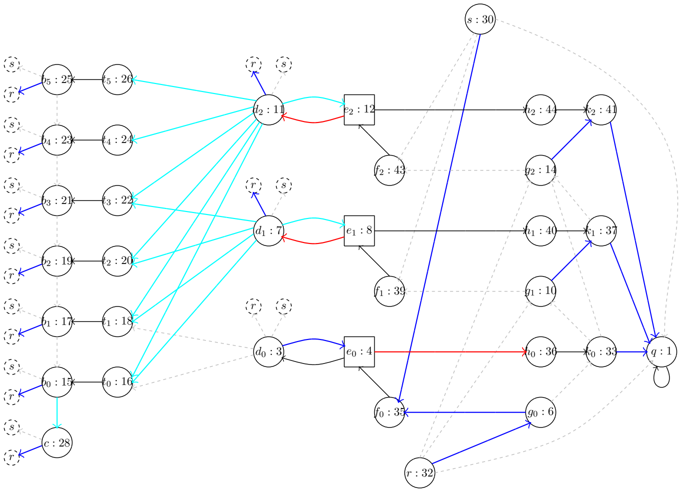
Deceleration lane and other cycles reset.

Full Construction



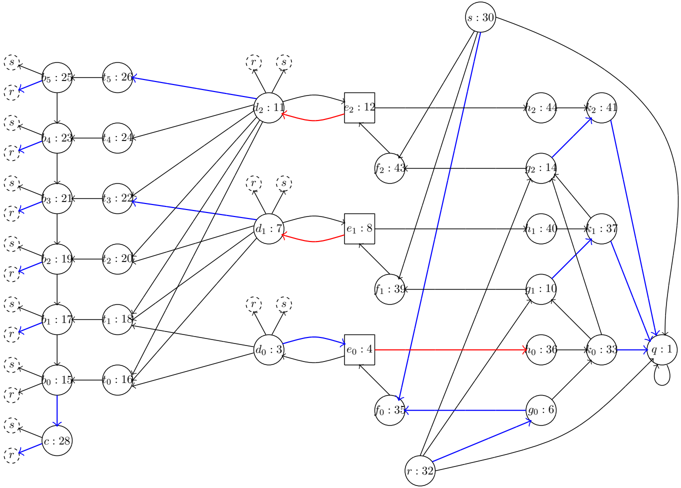
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



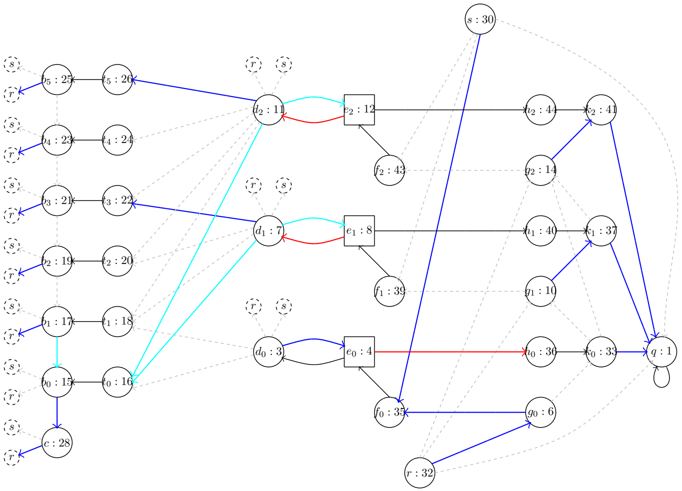
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



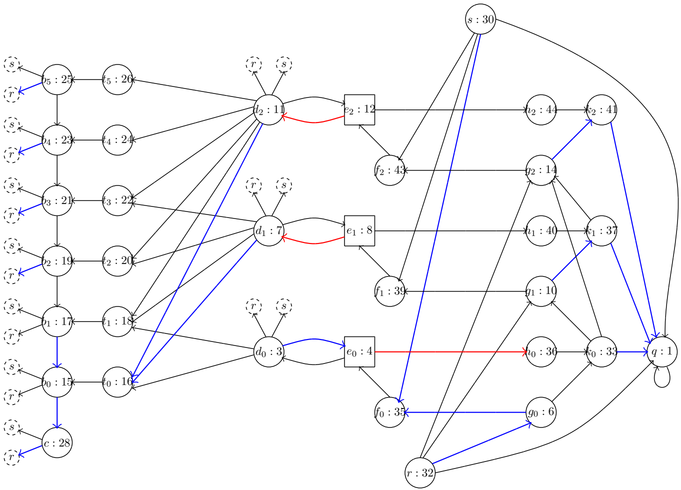
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



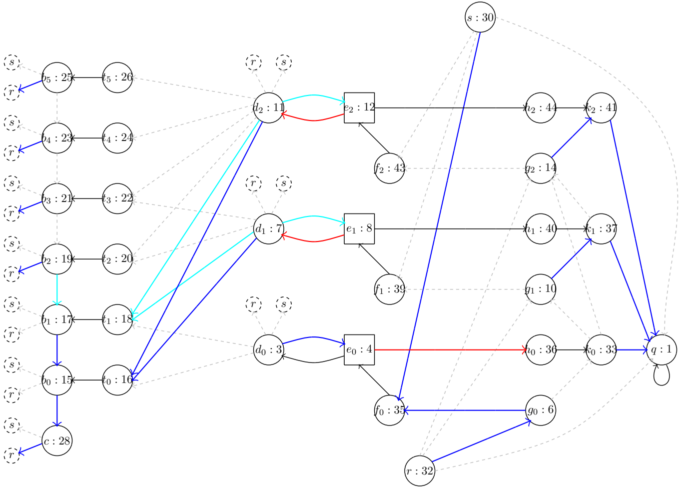
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



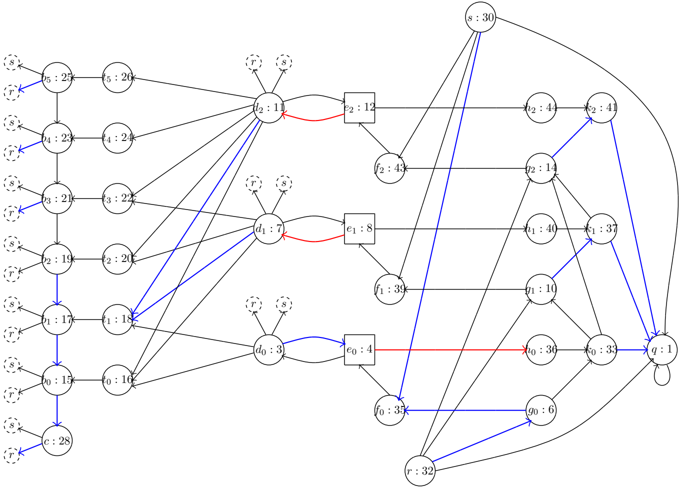
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



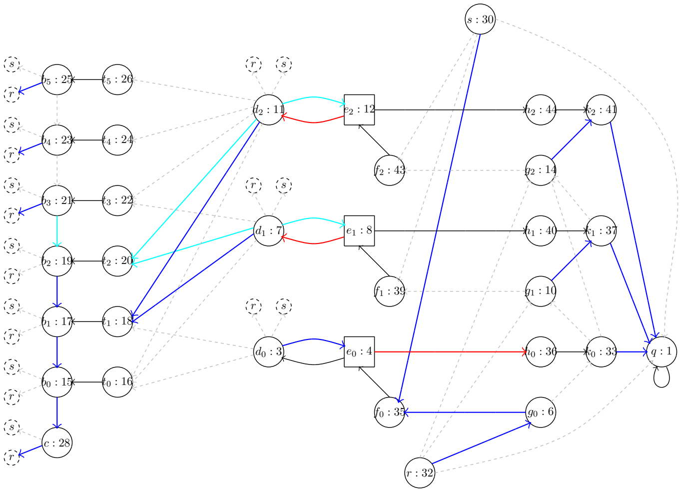
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



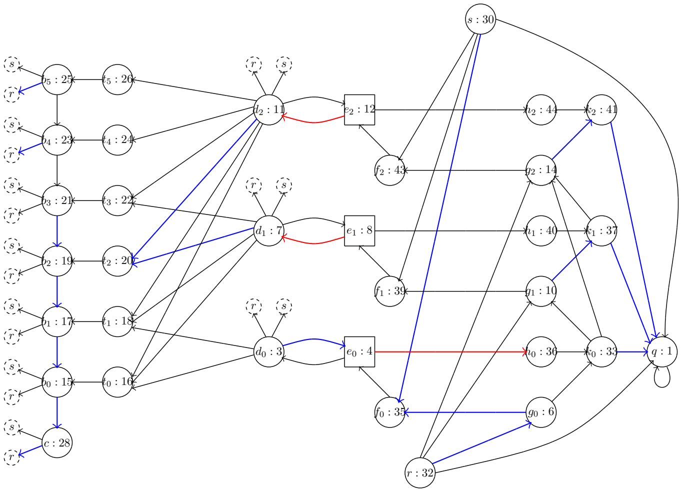
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



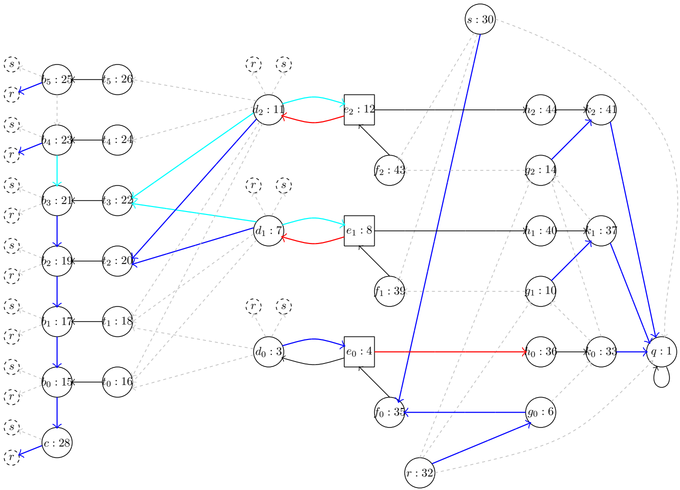
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



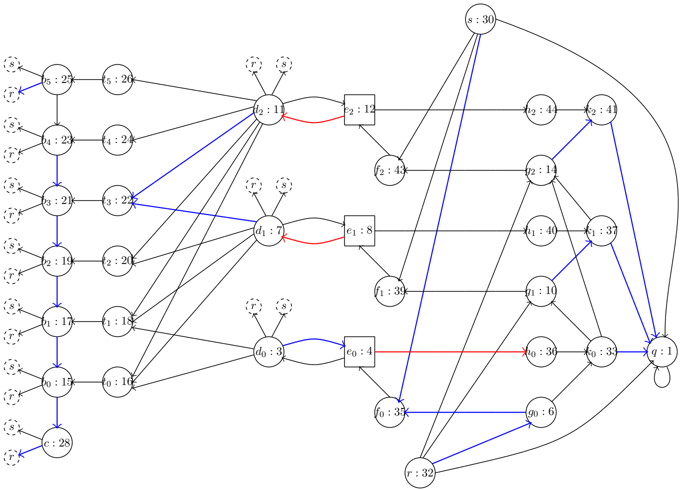
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



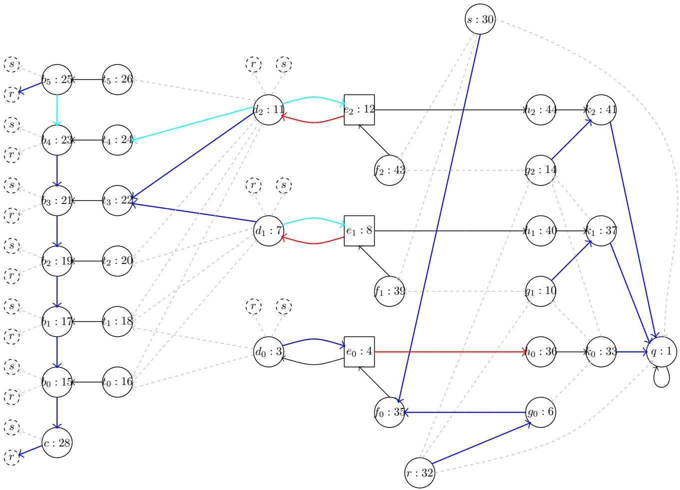
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



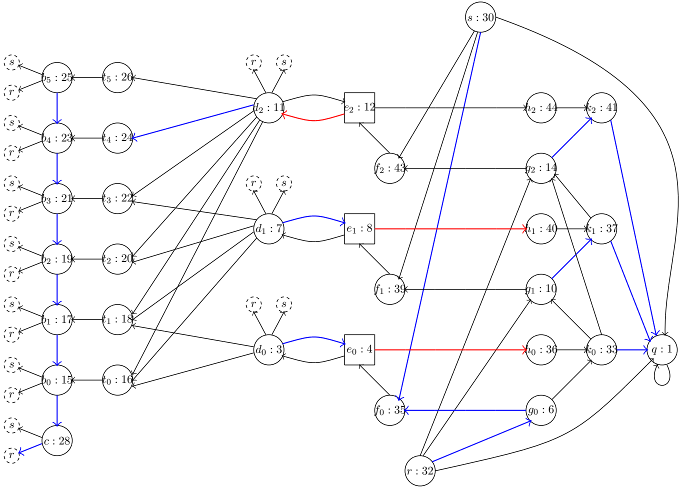
Lane improves iteratively, second and third cycle are occupied thereby.

Full Construction



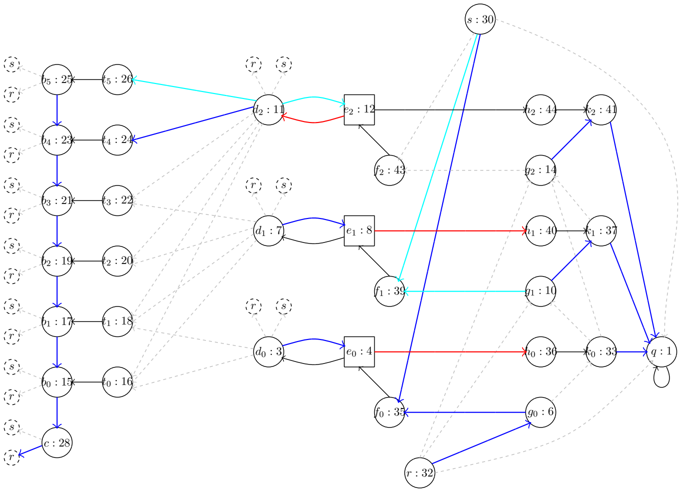
Second cycle cannot improve furthermore to the lane.

Full Construction



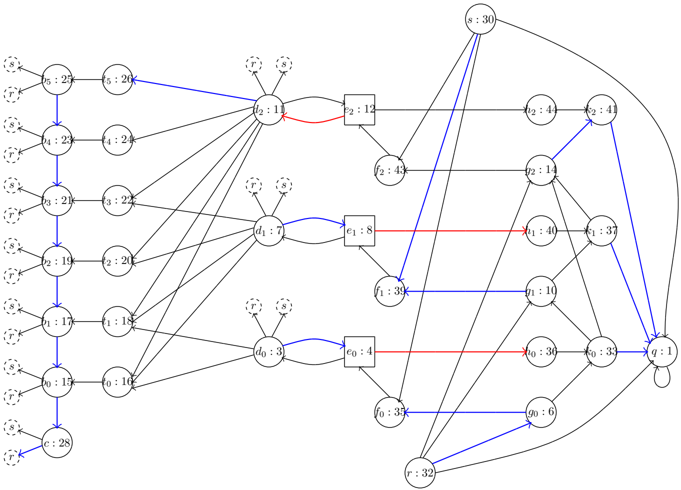
Second cycles closes, forcing player 1 to leave it.

Full Construction



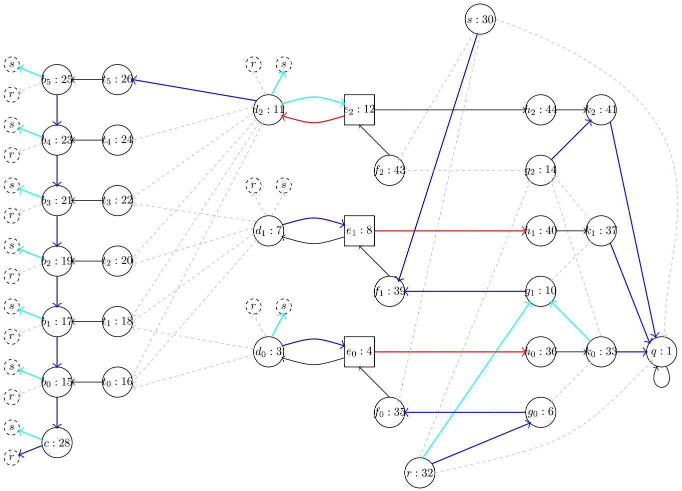
Second cycles closes, forcing player 1 to leave it.

Full Construction



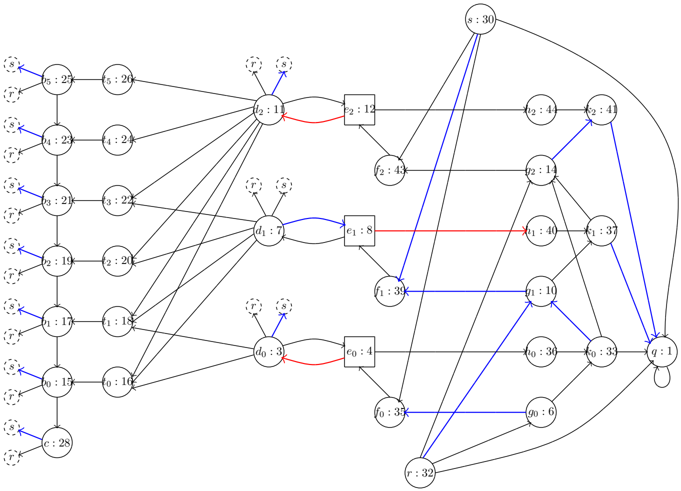
Second cycles closes, forcing player 1 to leave it.

Full Construction



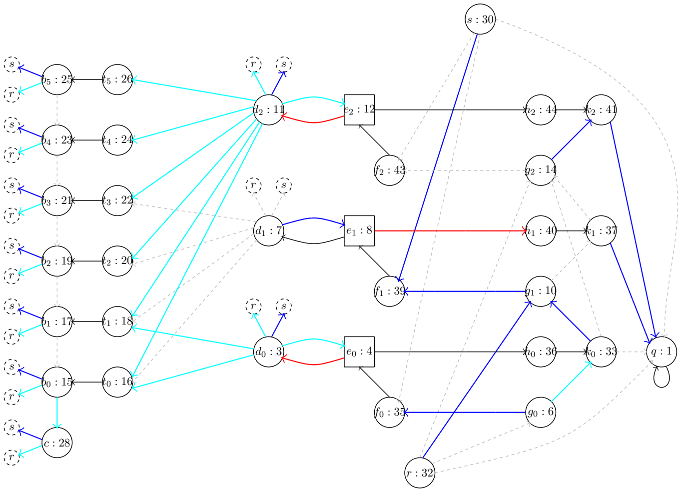
First cycle reopens again.

Full Construction



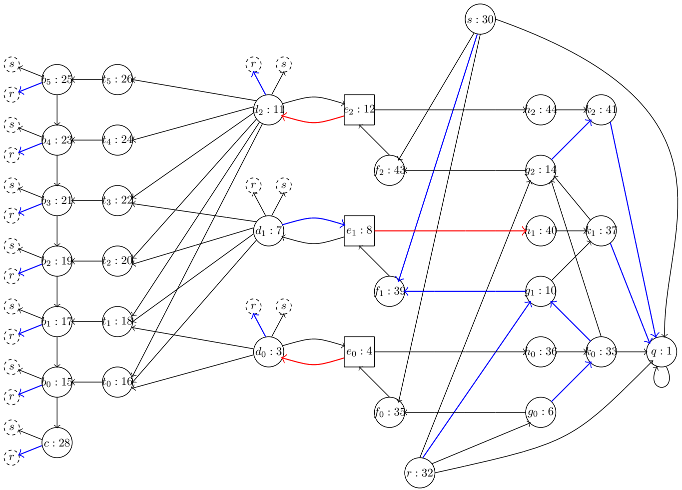
First cycle reopens again.

Full Construction



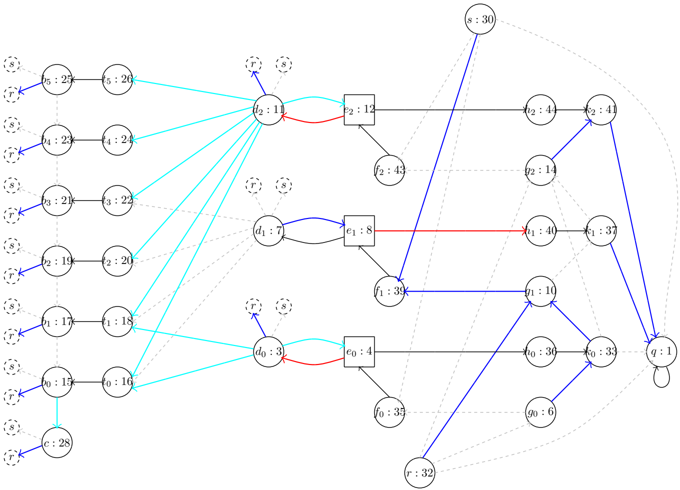
Deceleration lane and all other cycles reset.

Full Construction



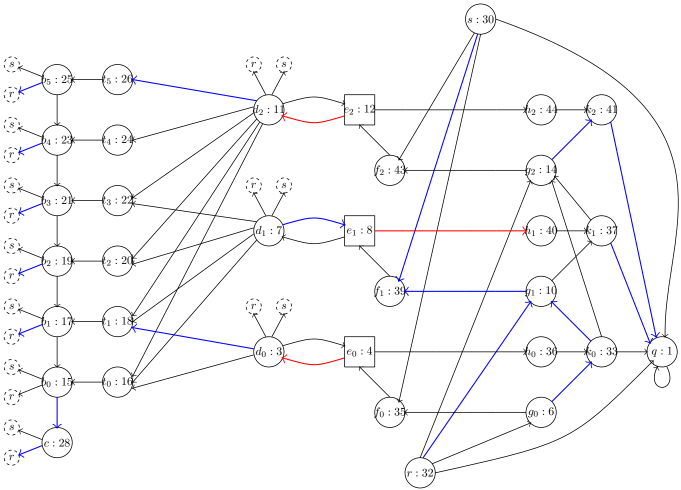
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



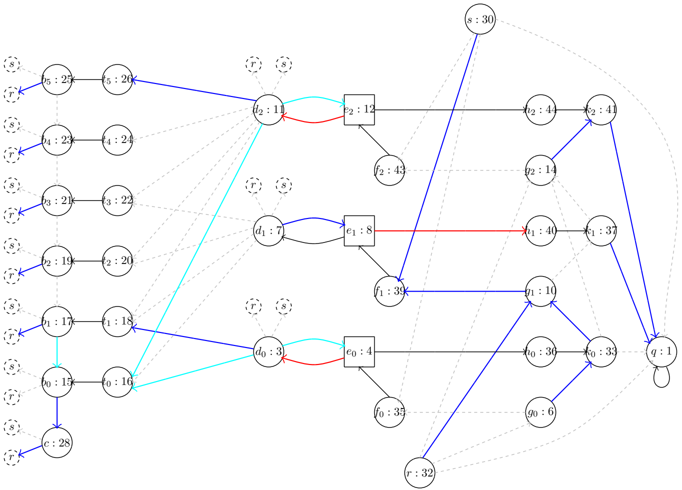
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



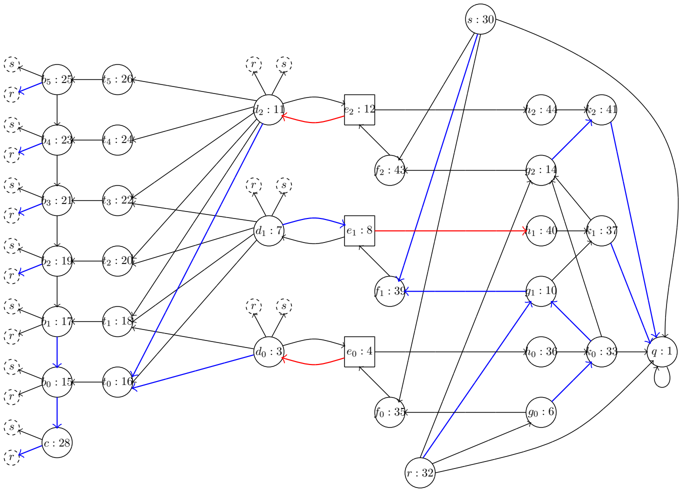
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



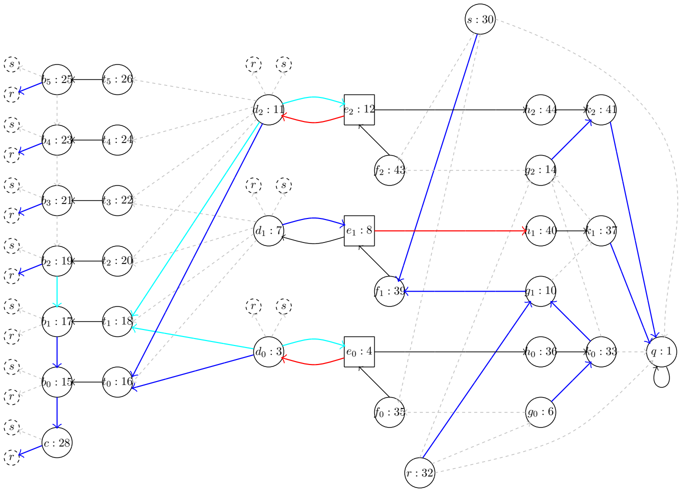
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



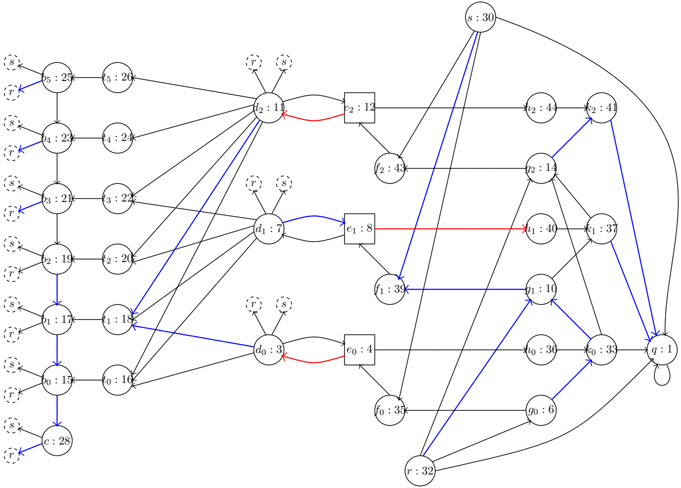
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



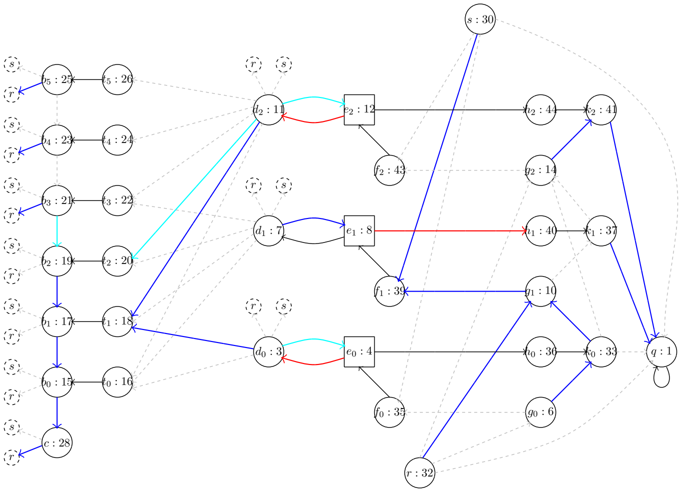
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



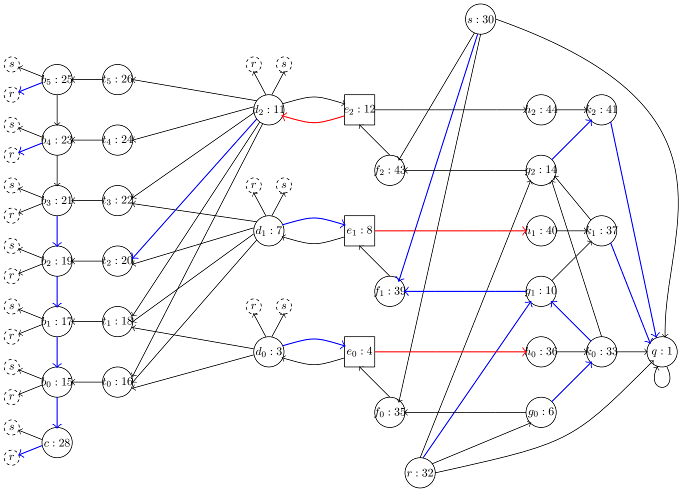
Lane improves iteratively, first and third cycle are occupied thereby.

Full Construction



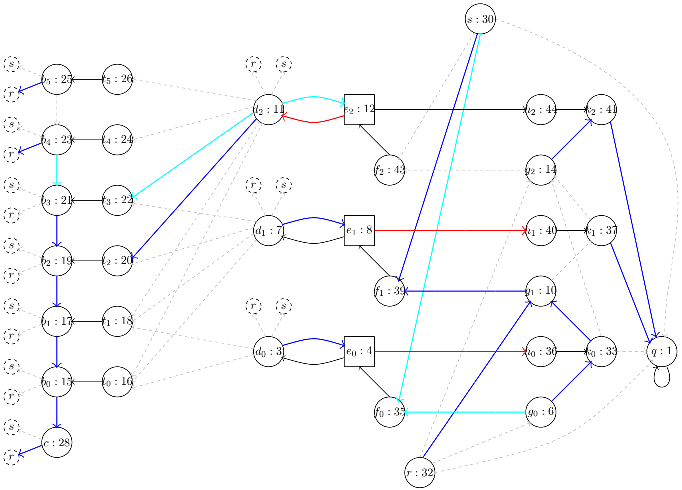
First cycle cannot improve furthermore to the lane.

Full Construction



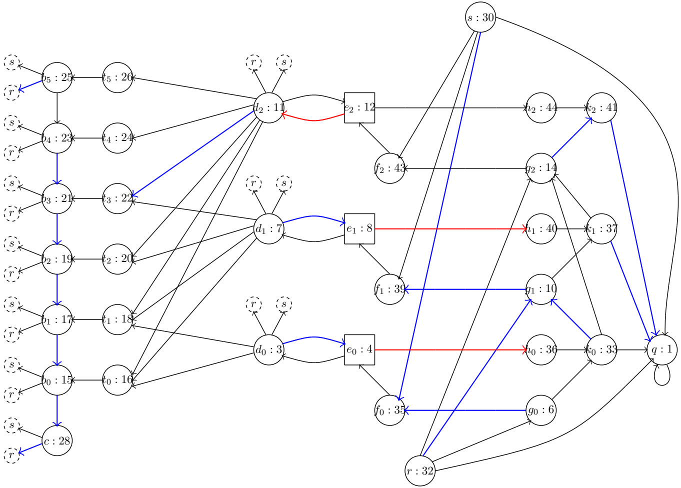
First cycles closes, forcing player 1 to leave it.

Full Construction



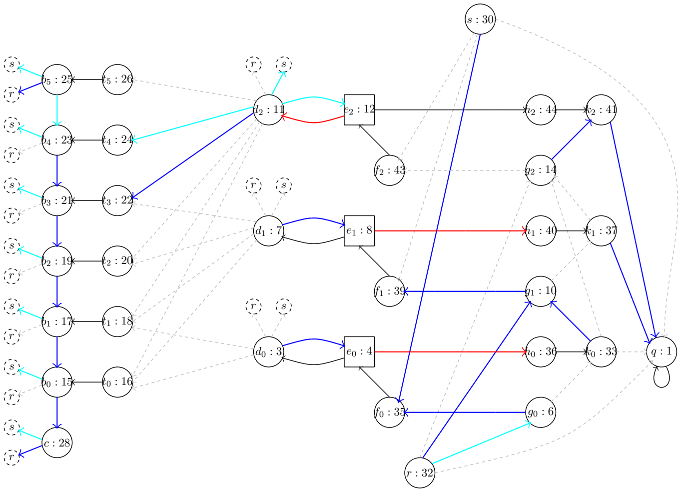
First cycles closes, forcing player 1 to leave it.

Full Construction



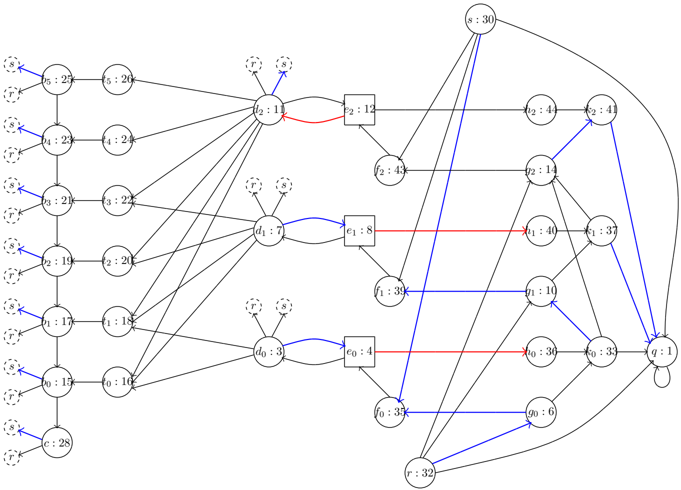
First cycles closes, forcing player 1 to leave it.

Full Construction



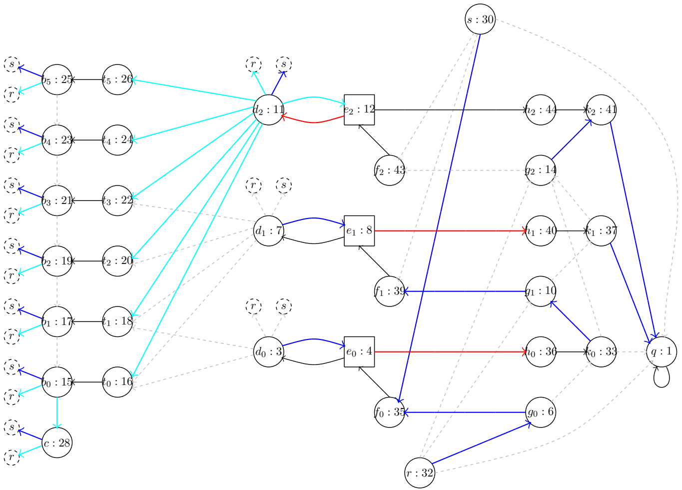
First cycles closes, forcing player 1 to leave it.

Full Construction



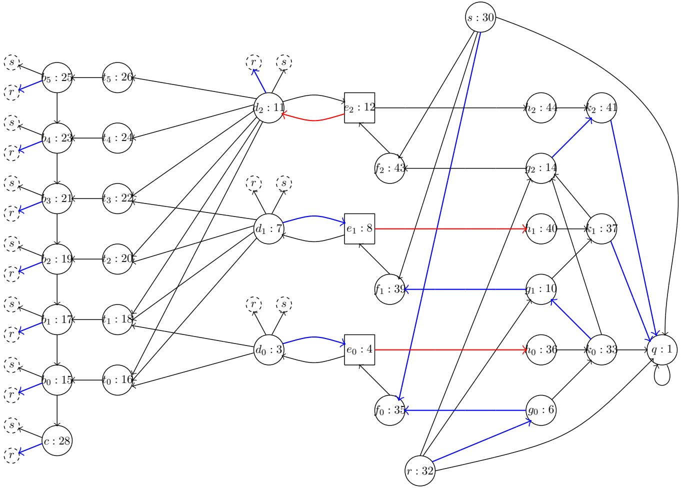
First cycles closes, forcing player 1 to leave it.

Full Construction



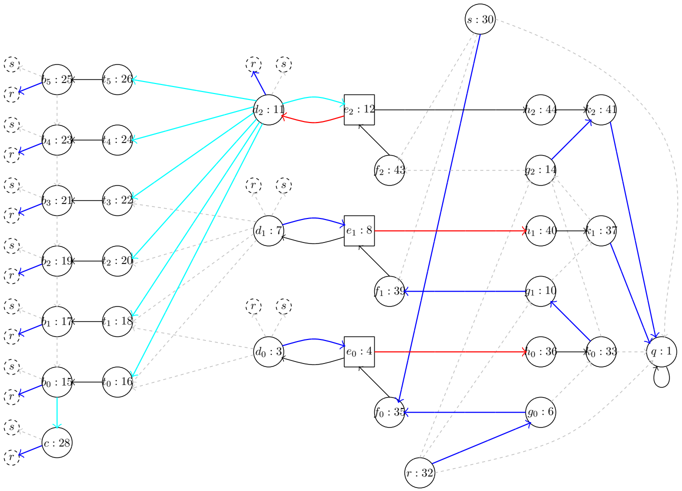
Deceleration lane and third cycle reset.

Full Construction



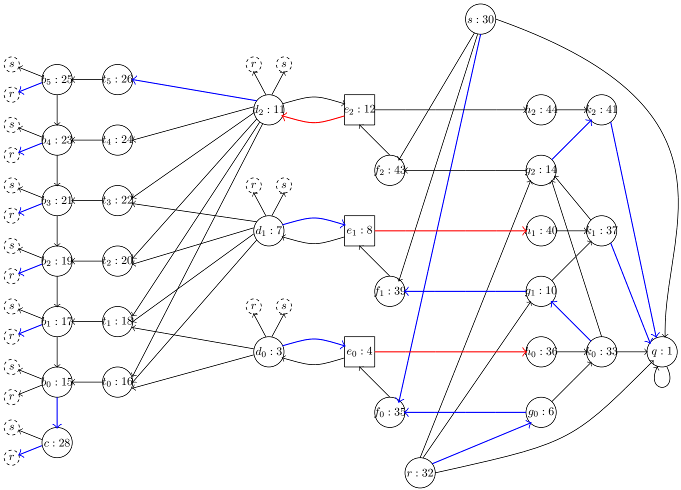
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



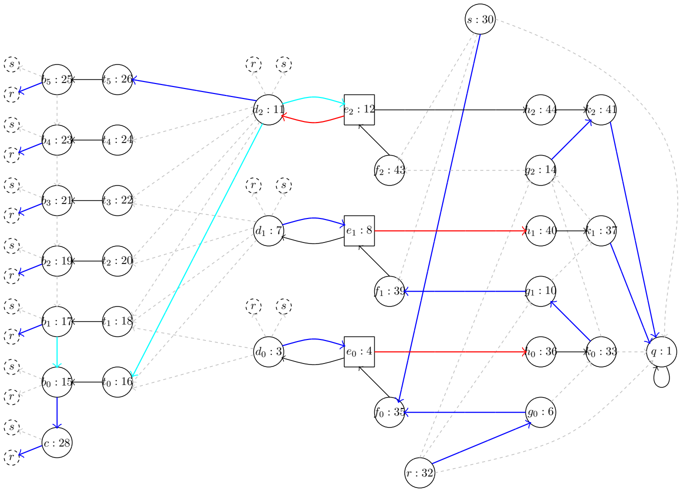
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



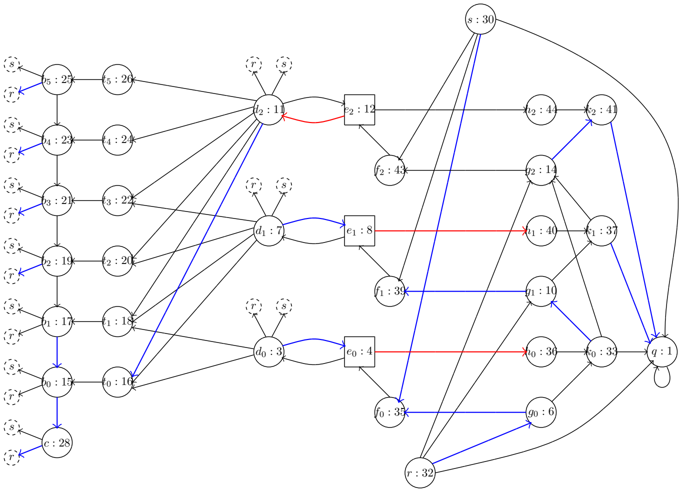
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



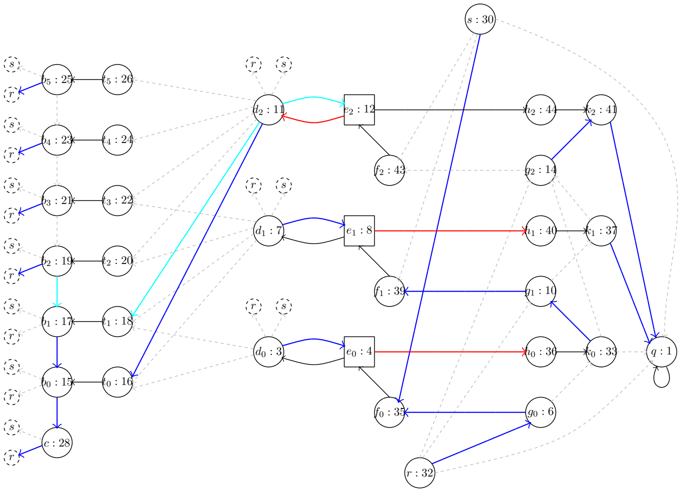
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



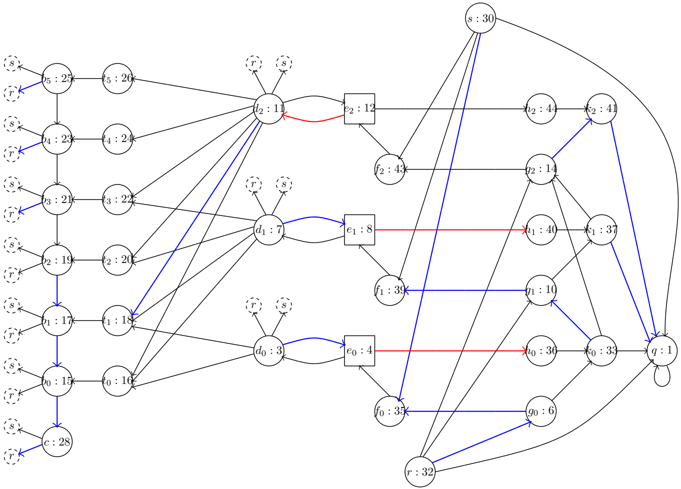
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



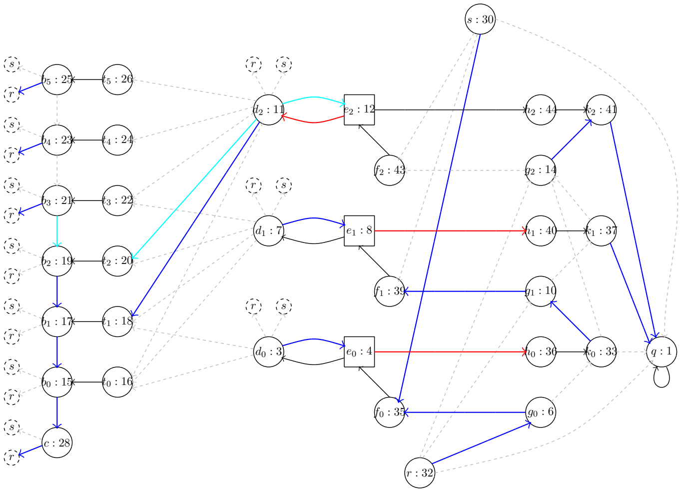
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



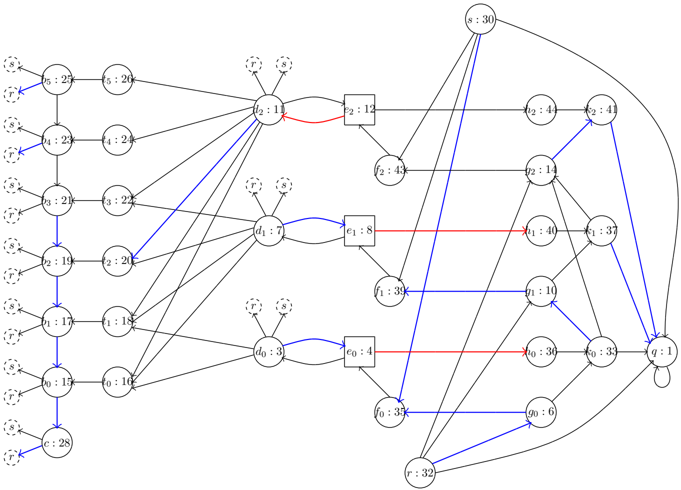
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



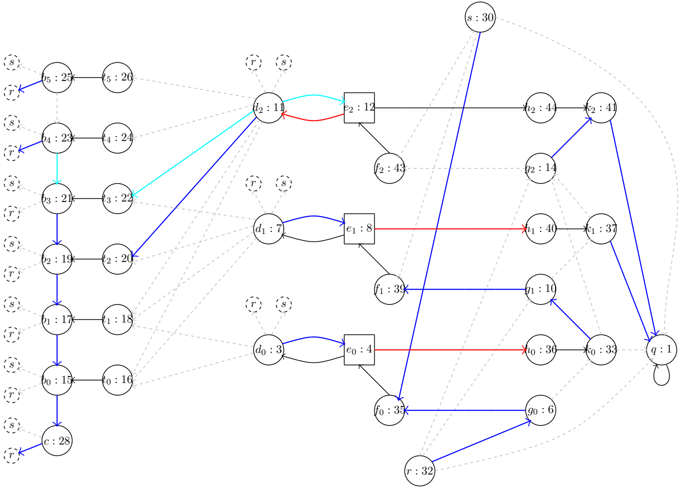
Lane improves iteratively, third cycle is occupied thereby.

Full Construction



Lane improves iteratively, third cycle is occupied thereby.

Full Construction



Lane improves iteratively, third cycle is occupied thereby.

Open problems

- Polytime algorithm for two-player games and the like