Steal Gold

(a.i) State representation

S = M x M x A x P x P x G

M = {-6, ...,-1} ∪ {1, ..., 6} # captured/active positions. Negative positions are used to indicate positions of captured agents

P = {1,.., 8} # players positions

G = {0, 1} # whether the gold has been stolen

A = {0, 1} # whether the alarm has been issued

(a.ii) Initial state and (a.iii) goal state

Initial: <3, 1, 0, 8, 8, 0> , **Goal**: <_, _, 0, _, _, 1>

World modeling

The environment evolves according to the following rules that are applied in the state resulting from the execution of an action of one of the players. We report them here, but they are not part of the requested solution.

- The effect of activating the alarm is to set a = 1. The alarm is issued if (1) one of the players is at the same position of an active policeman; or (2) two policemen are at the same node, one active and the other not.

  Alarm conditions: ((m0 = p0 **or** m0 = p1) **and** p0 ≠ p1)

  or ((m1 = p0 **or** m1 = p1) **and** p0 ≠ p1)

  or (|m0| = |m1| **and** (m0 > 0 **or** m1 > 0))
- The policemen are patrolling around the circle and their position m is updated by the following rule **m** ← **(m + 1)%7 + 1**.

  +1 for the next policeman position; mod(x, 7) for making the policeman walk in circle; and **+1** because the nodes start from 1.

Operators

Generic initial state for each action: <m0, m1, a, p0, p1, g>,
where $m0, m1 \in M$, $p0, p1 \in P$, $g \in G$, $a \in A$

GO(p, to) 'p' for the player and 'to' for the target position. The player ***p*** can move to ***to*** if (1) the alarm is not set; and (2) the next positions of both m0 and m1 are different from ***to*** or one of the policemen is at ***to*** but it's been captured; and (3) ***to*** is different from the current player position.

Pre :: a = 0
AND (($(m0 + 1)\%7 + 1 \neq to$ **AND** m0 > 0) **OR** m0 < 0)
AND (($(m1 + 1)\%7 + 1 \neq to$ **AND** m1 > 0) **OR** m1 < 0)
AND $p \neq to$
As effect the position of the player ***p*** is updated :: $p \leftarrow to$

GO-CAPTURE(p, m, other_p, other_m, to) 'p' for the player, 'm' for the policemen, 'Other_p' for the position of the other player, 'other_m' for the other policeman and 'to' for the target position. The player ***p*** can capture ***m*** and move to ***to*** if (1) the alarm is not set; (2) if active, the next position of ***other_m*** is different from ***to***; (3) the next position of ***m*** is equal to ***to***; (4) the other player ***other_p*** is already at ***to***; (5) the policeman to capture has to be active and the go action cannot happen in the current position of the player.
Pre :: a = 0
AND (($(other\_m + 1)\%7 + 1 \neq to$ **AND** other_m > 0) **OR** other_m < 0)
AND $(m + 1)\%7 + 1 = to$ **AND** $other\_p = to$
AND m0 > 0 **AND** $p \neq to$
As effect the position of the player ***p*** and the captured policeman ***m*** are updated: $p \leftarrow to$ and $m \leftarrow$ -to. $-to$,

STEAL(p) 'p' for the player. Policemen have to be captured and the alarm not set.
Pre :: m0 < 0 **AND** m1 < 0 **AND** a = 0
As effect the position of the player ***p*** is updated and the gold ***g*** is stolen: $p \leftarrow 7$ and $g \leftarrow 1$

One possible solution

1. GO(p0, 3) **->** s1: <4, 2, 0, 3, 8, 0>
2. GO-CAPTURE(p1, m1, p0, m0, 3) **->** s2: <5, -3, 0, 3, 3, 0>
3. GO(p0, 1) **->** s3: <6, -3, 0, 1, 3, 0>
4. GO-CAPTURE(p1, m0, p0, m1, 1) **->** s4: <-1, -3, 0, 1, 1, 0>
5. STEAL(p1) **->** s5: <-1, -3, 0, 1, 7, 1>

Steal Flags

(a.i) State representation

S = G0 x G1 x A x P x P x F x F

G0 = {-3, -2, -1} ∪ 1,2,3}	# guard0 positions. Negative positions are used to indicate positions of captured agents
G1 = {-5, -4, -3} ∪ {3,4,5}	# guard1 positions. Negative positions are used to indicate positions of captured agents
P = {1, …, 8}	# players positions
F = {0, 1}	# whether the flag has been stolen
A = {0, 1}	# whether he alarm has been issued

(a.ii) Initial state and (a.iii) goal state

Initial: <4, 1, 0, 6, 6, 0, 0> ,
Goal: <_, _, 0, _, _, 1, 1>

World modeling

The environment evolves according to the following rules that are applied in the state resulting from the execution of an action of one of the players. We report them here, but they are not part of the requested solution.

- The effect of activating the alarm is to set a = 1. The alarm is issued if (1) one of the players is at the same position of an active guard; or (2) two guards are at the same node, one active and the other not.

  Alarm conditions: ((g0 = p0 **or** g0 = p1) **and** p0 ≠ p1)
  or ((g1 = p0 **or** g1 = p1) **and** p0 ≠ p1)
  or (|g0| = |g1| **and** (g0 > 0 **or** g1 > 0))
- The guards are patrolling around the circle and their position g is updated by the following rules: **g** ← **(g + 1)%3 + 1** for g1, and **g** ← **((g-2) + 1)%3 + 3** for g0. **+1** for the next guard position; mod(x, 3) for making the guards walk in circle; and **+1, +3** because the nodes start from 1 for g1 and from 3 for g0.

Operators

Generic initial state for each action: <g0, g1, a, p0, p1, f0, f1>, where
$g0 \in G0,\ g1 \in G1,\ p0, p1 \in P,\ f0, f1 \in F,\ a \in A$

GO-C0(p, to) C0 is for the circle with the flag F0, ‘p’ for the player and ‘to’ for the target position. The player ***p*** can move to ***to*** if (1) the alarm is not set; (2) the next positions of both g0 and g1 are different from ***to*** or one of the guards at ***to*** it’s been captured; (3) the player position and target are in the same circle.

Pre :: a = 0

AND $(((g0-2)+1)\%3+3 \neq to$ **AND** g0 > 0) **OR** g0 < 0)

AND $((g1+1)\%3+1 \neq to$ **AND** g1 > 0) **OR** g1 < 0)

AND $to, p \in [1,2,3,8]$ // same circle condition

As effect the position of the player ***p*** is updated :: $p \leftarrow to$

The operator **GO-C1**(p, to) is equal to **GO-C0**(p, to) but the “same circle condition” which has to be substituted with $to, p \in [3,4,5,6,7]$

GO-CAPTURE-C0(p, other_p, to) ‘p’ for the player, ‘other_p’ for the other player, ‘to’ for the target position. The player ***p*** can capture ***g1*** in circle 0 and move to ***to*** if (1) the alarm is not set; (2) the next position of ***g1*** is equal to ***to***; (3) if active, the next position of the other guard is different from ***to***; (4) the player position and target are in the same circle; (5) ***other_p*** is already at ***to***.

Pre :: a = 0

AND $(g1+1)\%3+1 = to$ **AND** g1 > 0

AND $(((g0-2)+1)\%3+3 \neq to$ **AND** $g0 > 0)$ **OR** $g0 < 0)$

AND $to, p \in [1,2,3,8]$ // same circle condition

AND $other\_p = to$

As effect the position of the player ***p*** and the captured guard ***g1*** are updated: $p \leftarrow to$ and $g1 \leftarrow$-to.

GO-CAPTURE-C1(p, other_p, to) ‘p’ for the player, ‘other_p’ for the other player, and ‘to’ for the target position. The player ***p*** can capture ***g0*** in circle 1 and move to ***to*** if (1) the alarm is not set; (2) the next position of ***g0*** is equal to ***to***; (3) if active, the next position of the other guard is different from ***to***; (4) the player position and target are in the same circle; (5) ***other_p*** is already at ***to***.

Pre :: a = 0

AND $((g0-2)+1)\%3+3 = to$ **AND** g0 > 0

AND $((g1+1)\%3+1 \neq to$ **AND** $g1 > 0)$ **OR** $g1 < 0)$

AND $to, p \in [3,4,5,6,7]$ // same circle condition

AND $other\_p = to$

As effect the position of the player ***p*** and the captured guard ***g0*** are updated: $p \leftarrow to$ and $g0 \leftarrow$-to.

GO-CAPTURE-3(p, other_p, g, other_g) ‘p’ for the player, ‘other_p’ for the other player, and ‘g’ for a guard position and ‘other_g’ is for the other guard. The player ***p***

can capture ***g*** and move to ***3*** if (1) the alarm is not set; (2) the next position of ***g*** is **3**; (3) if active, the next position of ***other_g*** is different from ***3***; (4) ***other_p*** is at ***3***. This action is not mandatory to find a solution, but it can be defined to reduce its length.

Pre :: a = 0
AND (g = 1 **OR** g = 5)
AND ((other_g $\neq$ 1 **AND** other_g $\neq$ 5) **OR** other_g < 0)
AND other_p = 3

As effect the position of the player ***p*** and the captured guard ***g*** are updated: $p \leftarrow 3$ and $g0 \leftarrow -3$.

STEAL-F0(p) 'p' for the player. Both guards have to be captured, the alarm not set and they have to go to the flag position from the circle where the flag is located.

Pre :: g0 < 0 **AND** g1 < 0 **AND** a = 0
AND $p \in [1, 2, 3, 8]$ // same circle condition

As effect the position of the player ***p*** is updated and the flag ***f0*** is stolen: $p \leftarrow 8$ and $f0 \leftarrow 1$

The operator **STEAL-F1**(p) is equal to **STEAL-F0**(p) but the "same circle condition" which has to be substituted with $p \in [3, 4, 5, 6, 7]$

One possible solution

GO-C1(p0, 6)	-> s2: <5, 3, 0, 6, 6, 0, 0> // or a wait operator
GO-C1(p0, 4)	-> s3: <3, 2, 0, 4, 6, 0, 0>
GO-CAPTURE-C1(p1, p0, 4)	-> s4: <-4, 1, 0, 4, 4, 0, 0>
GO-C1(p0, 4)	-> s5: <-5, 3, 0, 4, 4, 0, 0> // or a wait operator
GO-C1(p0, 3)	-> s6: <-5, 2, 0, 3, 4, 0, 0>
GO-C1(p0, 3)	-> s7: <-5, 1, 0, 3, 3, 0, 0> // or a wait operator
GO-CAPTURE-3(p1, p0, g1, g0)	-> s8: <-5, -3, 0, 3, 3, 0, 0>
STEAL-F1(p0)	-> s9: <-5, -3, 0, 7, 3, 0, 1>
STEAL-F0(p1)	-> s10: <-5, -3, 0, 7, 8, 1, 1>