The Complexity of Speedrunning Video Games

Manuel Lafond

U of Ottawa/U of Sherbrooke

Speedrunning

- No time to play video games anymore…

- => Just watch people play then.

Speedrunning

- No time to play video games anymore…
- => Just watch people play then.
- Sounds boring?

Speedrunning

- No time to play video games anymore…
- => Just watch people play then.
- Sounds boring?
- => Speedrunning makes it interesting. (at least for me)

Super Mario Bros.
any% 21183

1-1 (Today: 35)	-0.3
1-2	-0.3
4-1 (10)	-
4-2	-0.3
8-1	-0.3
8-2	-0.3
8-3	-0.3
King Koopa	-0.2

158 4:57.02

cyghfer: gg
xx_420_blazit_xx: HOOOOWSFDSFDSFDS
[illegible]
[illegible] 456 456 456
macehead_15:

MARIO 096700 ×46 WORLD 8-4 TIME 315

THANK YOU MARIO!

YOUR QUEST IS OVER.
WE PRESENT YOU A NEW QUEST.

PUSH BUTTON B
TO SELECT A WORLD

Speedrunning

- **Goal:** finish a game as fast as possible.

- Very *competitive* field, but also very *collaborative*.
- Standard speedrunning techniques developed over the years.
  - In this talk:
    - **Part 1: damage boosting**
    - **Part 2: routing stages**
- Lead to algorithmic problems.
  - Damage boosting => generalization of knapsack
  - Routing stages => generalization of feedback arc set

Damage boosting

(incorporation of multimedia content!)

Some related work

- For many games *played by speedrunners*, it is NP-hard/PSPACE-hard to decide if the game can be completed AT ALL…
  - Lemmings *[Cormore, 2004]*
  - Super Mario Bros, Donkey Kong Country, Zelda *[Aloupis & al., 2015]*
  - Many, many more: meta-theorems of *[Viglietta, 2014]*

- *Mario Kart problem*: can a given course be finished in k seconds? *[Bosboom & al., 2015]*

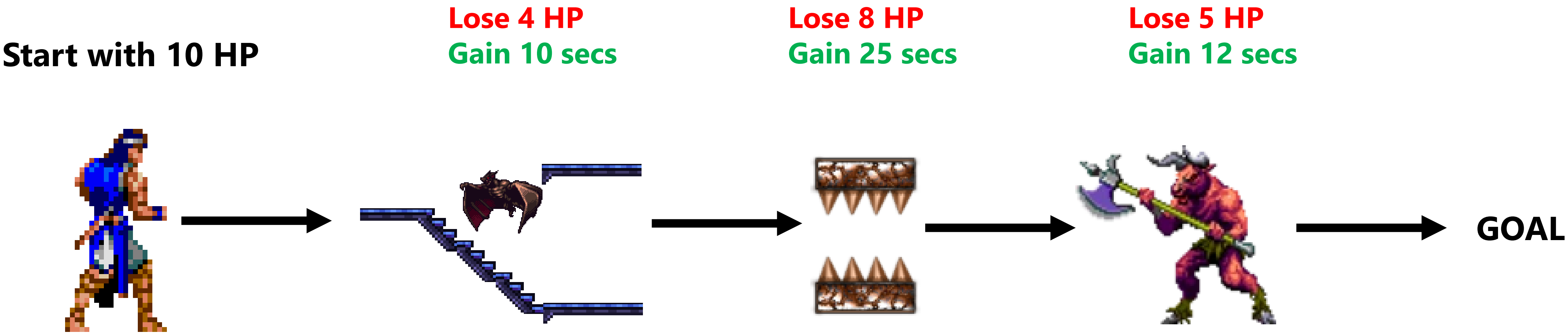
Some related work

- Speedrunners often measure **time gained w.r.t « normal play »**.
  - e.g. save 40 seconds by damage-boosting on the bat.

- In this work, a game is a set or a sequence of **time-gaining events** (opportunities to gain time that can be taken or not).
  - Goal: maximize total time-gain on these events.
  - This formulation avoids problem of **unfinishable games**.
  - Approximation algorithms
  - Fixed-parameter tractability

Damage boosting

Damage boosting through a stage



Start with 10 HP

Lose 4 HP
Gain 10 secs

Lose 8 HP
Gain 25 secs

Lose 5 HP
Gain 12 secs

GOAL

Damage boosting through a stage

Start with 10 HP

Lose 4 HP
Gain 10 secs

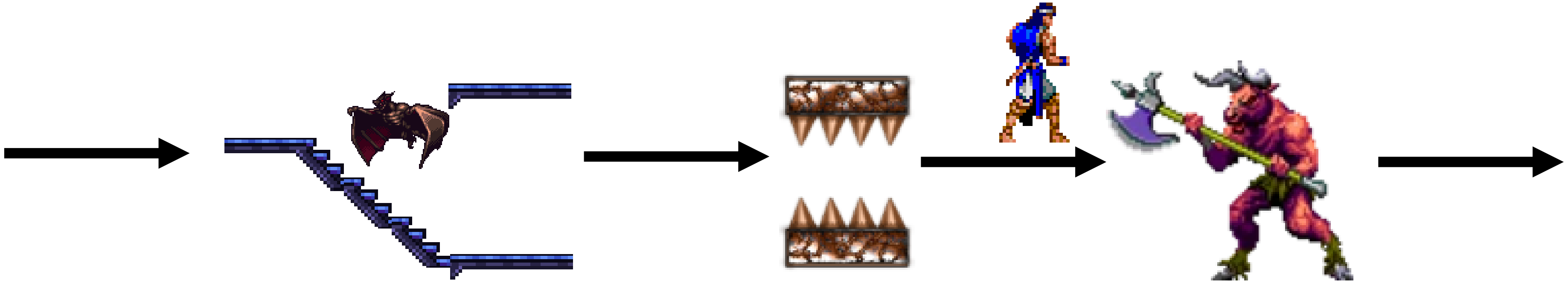
6 HP

Lose 8 HP
Gain 25 secs

Lose 5 HP
Gain 12 secs

GOAL

Damage boosting through a stage



Start with 10 HP

Lose 4 HP
Gain 10 secs

6 HP

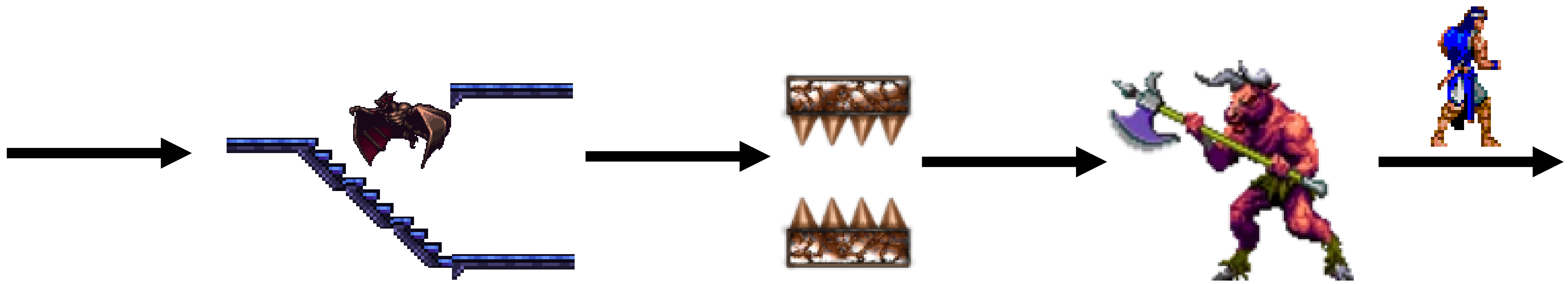
~~**Lose 8 HP**~~
~~**Gain 25 secs**~~

6 HP

Lose 5 HP
Gain 12 secs

GOAL

Damage boosting through a stage



Start with 10 HP

Lose 4 HP
Gain 10 secs

6 HP

~~Lose 8 HP~~
~~Gain 25 secs~~

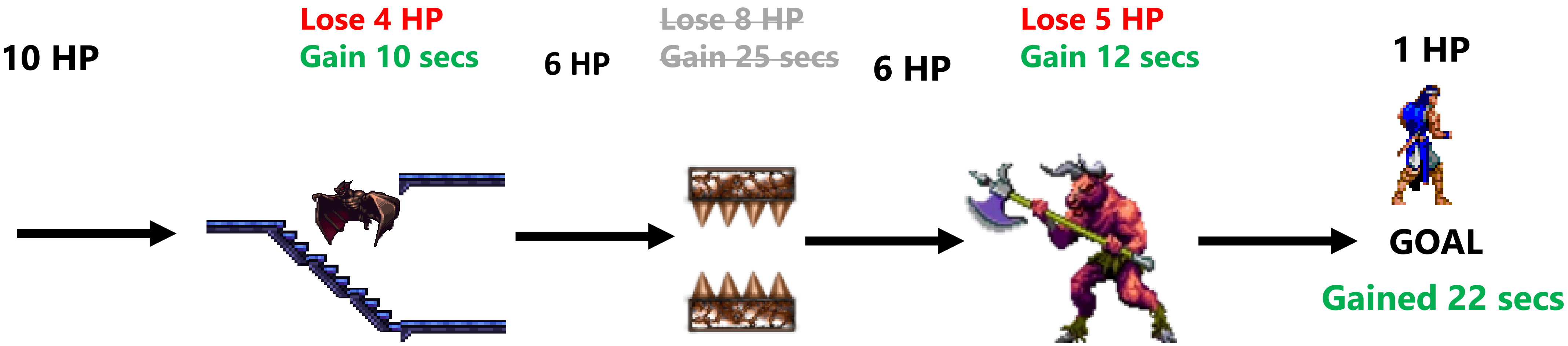
6 HP

Lose 5 HP
Gain 12 secs

1 HP

GOAL

Damage boosting through a stage



Start with 10 HP

Lose 4 HP
Gain 10 secs

6 HP

~~Lose 8 HP~~
~~Gain 25 secs~~

6 HP

Lose 5 HP
Gain 12 secs

1 HP

GOAL

Gained 22 secs

Damage boosting through a stage

Start with 10 HP

~~Lose 4 HP~~
~~Gain 10 secs~~

Lose 8 HP
Gain 25 secs

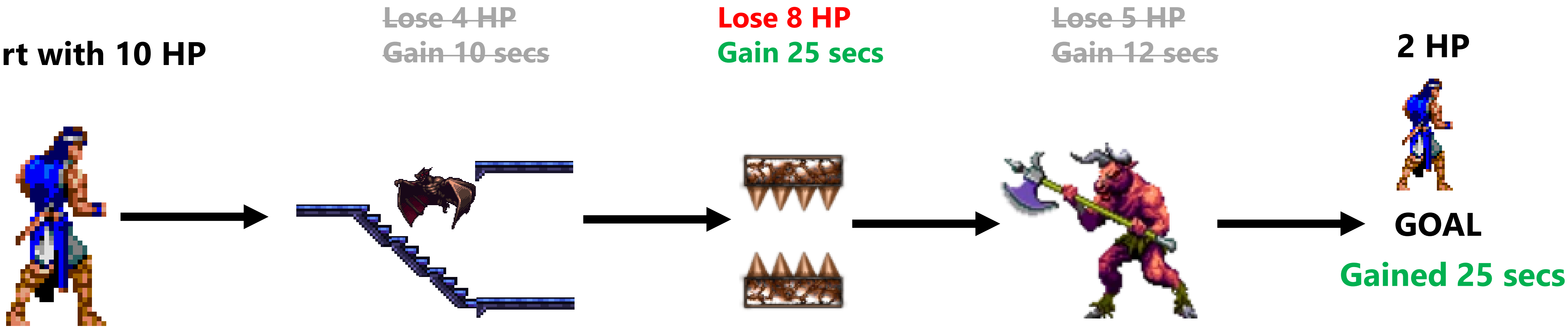
~~Lose 5 HP~~
~~Gain 12 secs~~

2 HP

GOAL

Gained 25 secs

Damage boosting through a stage



Start with 10 HP

~~**Lose 4 HP**~~
~~**Gain 10 secs**~~

Lose 8 HP
Gain 25 secs

~~**Lose 5 HP**~~
~~**Gain 12 secs**~~

2 HP

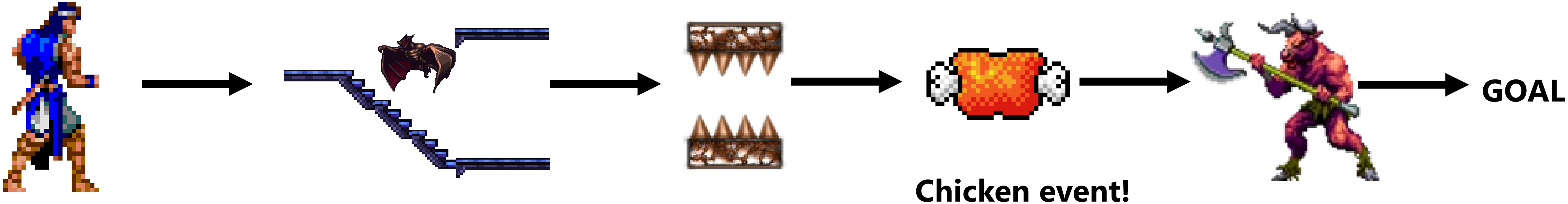
GOAL

Gained 25 secs

Note: this is the knapsack problem.
Maximize time gains without spending more than max HP =
Maximize value of items while staying under maximum weight.


11

Chicken events



Start with 10 HP

Lose 4 HP
Gain 10 secs

Lose 8 HP
Gain 25 secs

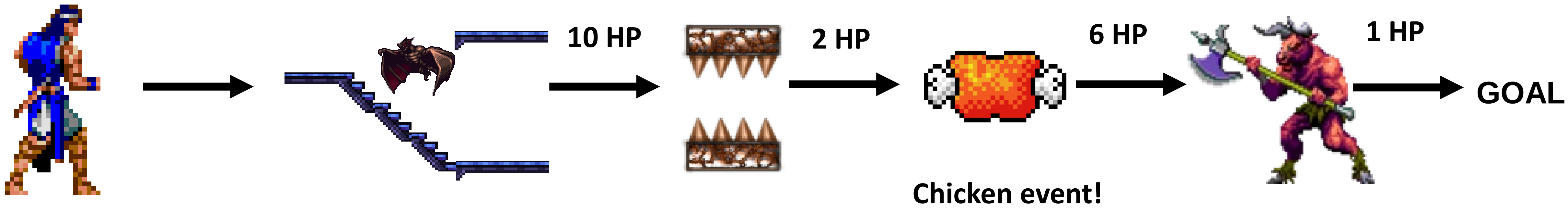
Gain 4 HP
Lose 8 secs

Chicken event!

Lose 5 HP
Gain 12 secs

GOAL

Chicken events



Start with 10 HP

~~Lose 4 HP~~
~~Gain 10 secs~~

Lose 8 HP
Gain 25 secs

Gain 4 HP
Lose 8 secs

Lose 5 HP
Gain 12 secs

10 HP

2 HP

6 HP

1 HP

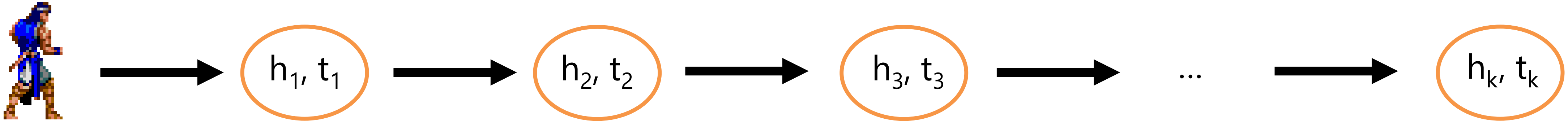
GOAL

Chicken event!

The Damage Boosting problem

- **Given**: a sequence of events $S = ((h_1, t_1), \dots, (h_k, t_k))$ and starting hit points hp.
  - h_i is the HP lost and t_i the time gained if event (h_i, t_i) is taken.
  - Both values are negative for chicken events.

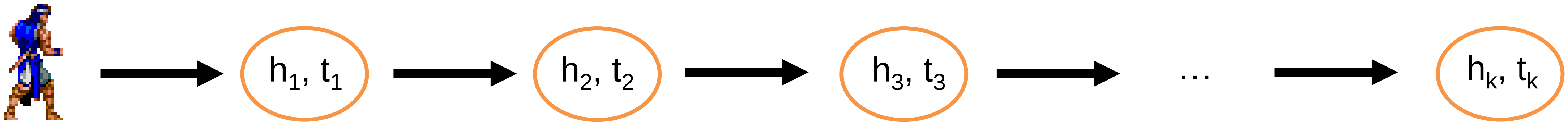
- **Find**: a subsequence S' of S of events to take such that
  - The player hp is always strictly above 0.
  - The sum of time gains is maximized.



The Damage Boosting problem

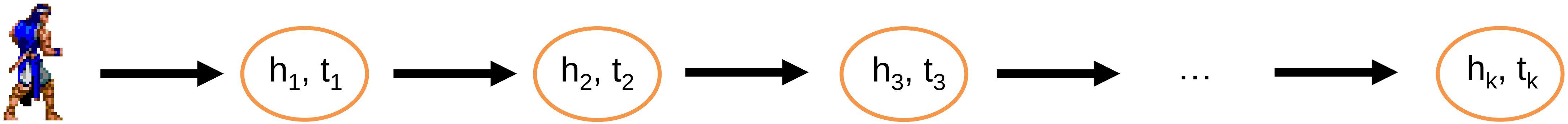
- **Given**: a sequence of events $S = ((h_1, t_1), \ldots, (h_k, t_k))$ and starting hit points hp.
  - h_i is the HP lost and t_i the time gained if event (h_i, t_i) is taken.
  - Both values are negative for chicken events.

- **Find**: a subsequence S' of S of events to take such that
  - The player hp is always strictly above 0.
  - The sum of time gains is maximized.



A PTAS for damage boosting

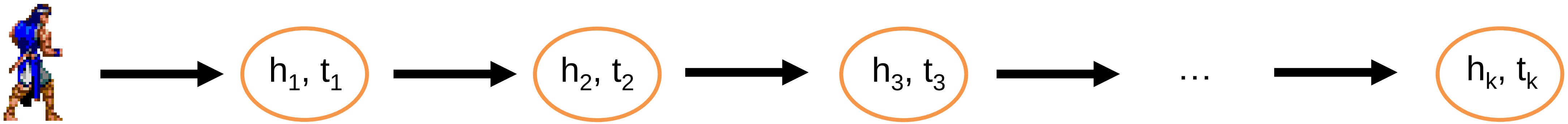
- The polynomial-time approximation scheme (PTAS) for knapsack still works with minor modifications.
  - Pseudo-polynomial **dynamic programming** algorithm running in time $O(n^2T)$ n = *number of events* T = *max time gain*
  - Scale the time gains by $\varepsilon T/n$, run the DP algorithm, get a solution of value at least $(1 - \varepsilon)$ OPT.



h_1, t_1 → h_2, t_2 → h_3, t_3 → … → h_k, t_k

FPT of damage boosting

- In practice, the h_i's should not take too many possible values:
  - Each enemy does a **fixed amount of damage**, and a game usually has **few enemy types**.
- Knapsack is FPT in the **number of distinct weight values** present in the input.
  - If k possible weights, can be solved in time $O(2^{2.5k\log k} poly(n))$.



h_1, t_1

h_2, t_2

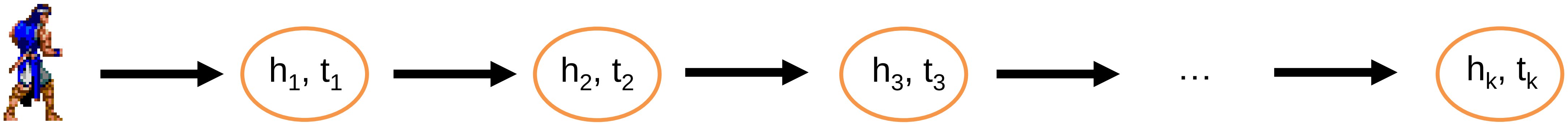
h_3, t_3

…

h_k, t_k

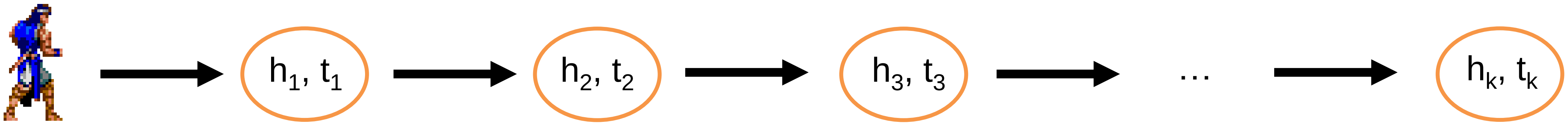
FPT of damage boosting

- **Question**: if *k* is the number of possible damage values, is damage boosting FPT in *k* ?



FPT of damage boosting

- **Question**: if *k* is the number of possible damage values, is damage boosting FPT in *k* ?
  - **Answer: I don’t know…**
  - FPT in ***c* + *k***, where *c* is the number of chicken events.
  - $O(2c(2k(c+1) + c)^{2.5(2k(c+1) + c)}poly(n))$ algorithm.
    - Involves ILP with $O(c + k)$ variables, using results of *[Lokshtanov, 2009]*



h_1, t_1 → h_2, t_2 → h_3, t_3 → … → h_k, t_k

FPT of damage boosting

k = number of distinct damage values

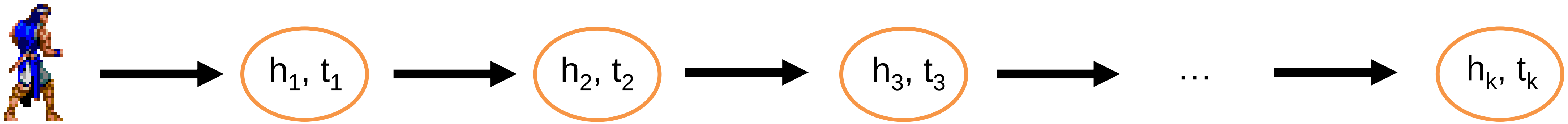
r = number of chicken events taken

$$\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{i=1}^{k}\sum_{j=0}^{r} g_{ij} & \\
\text{subject to} & h_{j+1} \leq h_j - \displaystyle\sum_{i=1}^{k} x_{ij} d_i - d(c_{j+1}) & j \in \{0, \ldots, r-1\} \\
& h_j \leq hp & j \in \{1, \ldots, r\} \\
& h_j - \displaystyle\sum_{i=1}^{k} x_{ij} d_i > 0 & j \in \{0, \ldots, r\} \\
& g_{ij} \leq f_{ij}(x_{ij}) & i \in [k], j \in \{0, \ldots, r\} \\
& h_j \in \mathbb{N} & j \in \{1, \ldots, r\} \\
& x_{ij} \in \{0, \ldots, n_{ij}\}, g_{ij} \in \mathbb{N} & i \in [k], j \in \{0, \ldots, r\}
\end{array}$$

h_j = HP remaining after taking j-th chicken

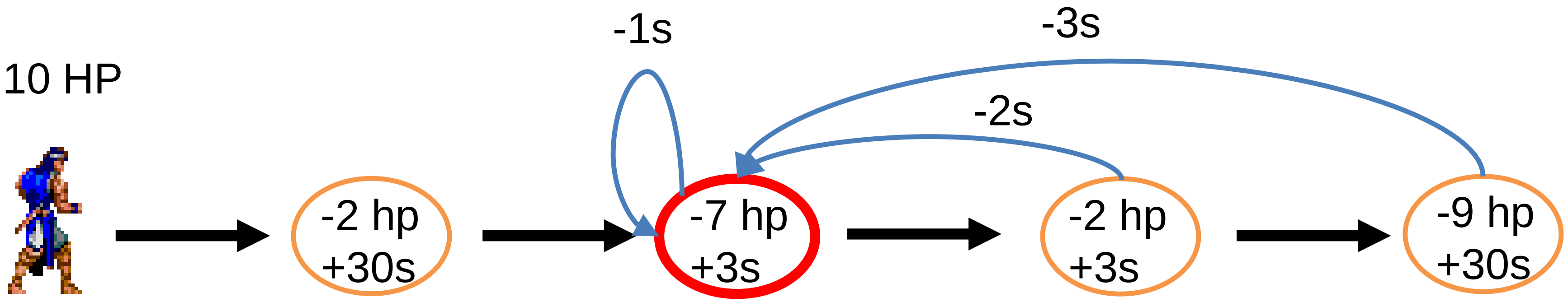
x_{ij} = number of events of damage value d_i taken between chicken *j* and *j+1*

g_{ij} = time gained from damage d_i events between chicken *j* and *j+1*



Also in the paper

- Damage boosting with multiple lives.
- Allow HP to drop to 0.
  - Lose a life = respawn at last checkpoint **with full HP**
  - Limited number of lives *L*
  - Maximum time gain is hard to approximate within factor ½
  - Pseudo-polynomial time algorithm $O(n^2 (maxHP)^2 L)$.



Routing stages

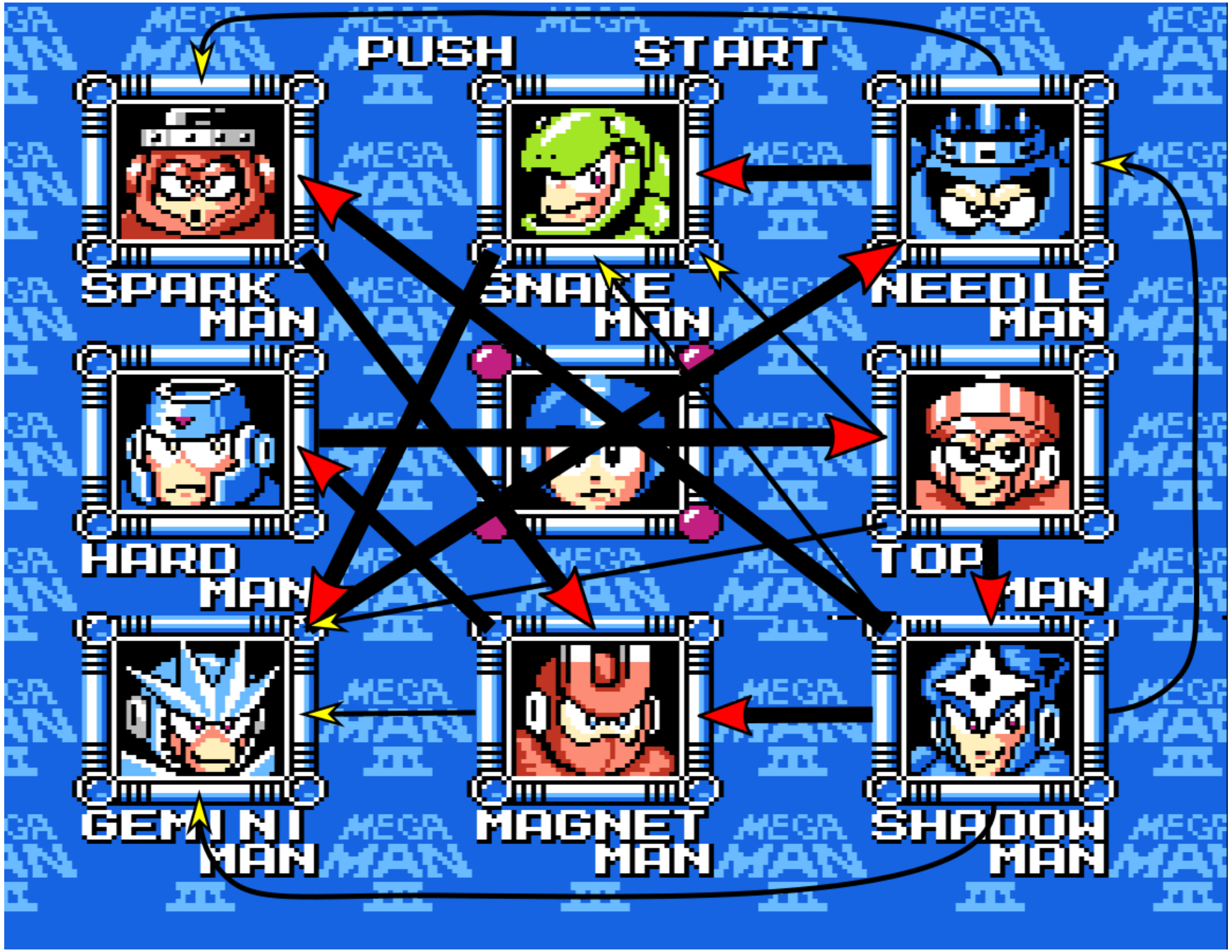
Routing


PUSH START
SPARK MAN
SNAKE MAN
NEEDLE MAN
HARD MAN
TOP MAN
GEMINI MAN
MAGNET MAN
SHADOW MAN
MEGA MAN III


MAGNET
MAN


YOU GOT
MAGNET MISSILE

Routing

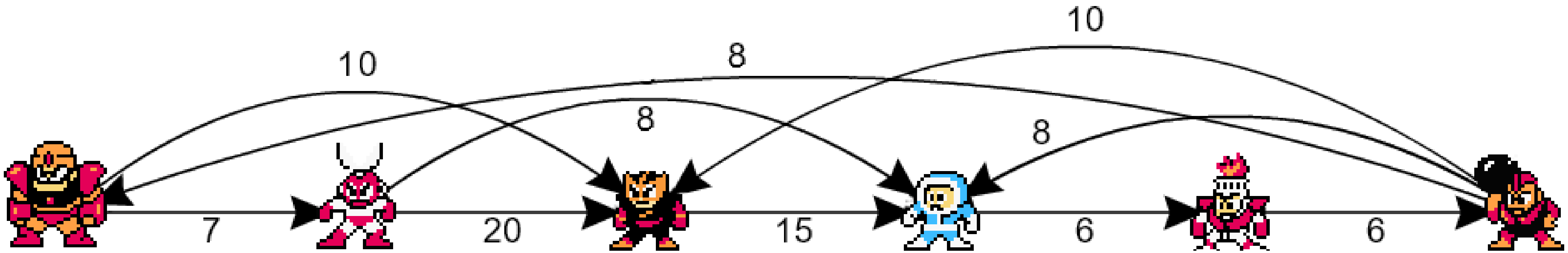


PUSH START
SPARK MAN
SNAKE MAN
NEEDLE MAN
HARD MAN
TOP MAN
GEMINI MAN
MAGNET MAN
SHADOW MAN

The Stage Routing problem

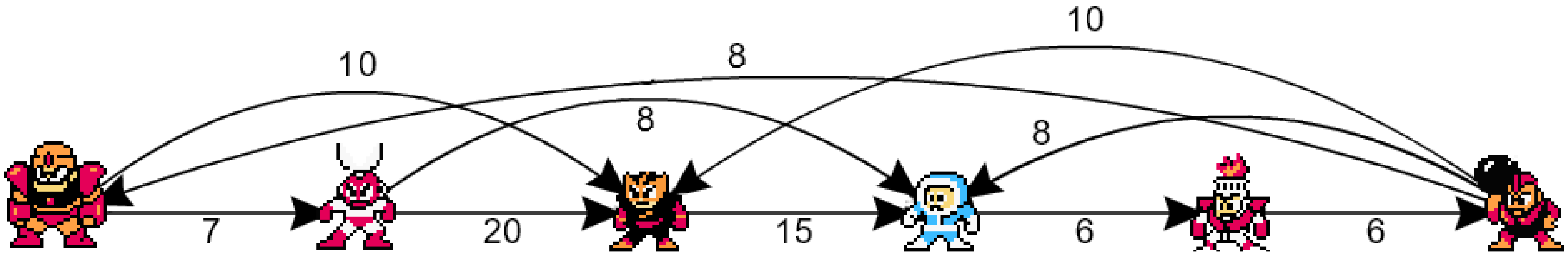
- **Given**: a set of stages $S = \{S_1, \ldots, S_k\}$ in which the time to complete S_i depends on the weapons acquired from the stages completed before.

- **Find**: a completion order of S that maximizes time gain.



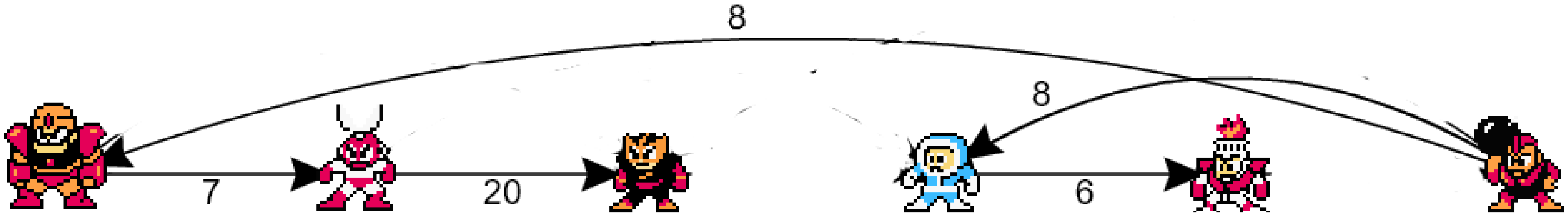
The Stage Routing problem

- Below: a stage is just a boss.
- Find a max-weight acyclic sub-digraph of indegree at most 1.
  - Gives ordering + which weapons beats which boss.



The Stage Routing problem

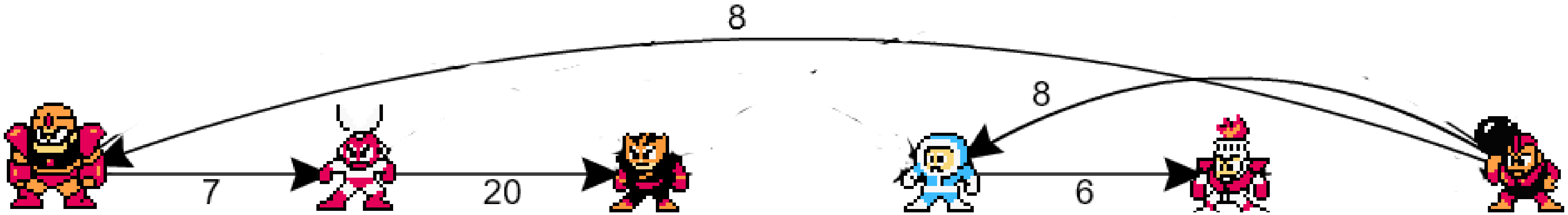
- Below: a stage is just a boss.
- Find a max-weight acyclic sub-digraph of indegree at most 1.
  - Gives ordering + which weapons beats which boss.
  - Called an **arborescence**, found in time *O(|A| + n log n) [Gabov & al., 1986]*.



The Stage Routing problem

- Below: a stage is just a boss.
- Find a max-weight acyclic sub-digraph of indegree at most 1.
    - Gives ordering + which weapons beats which boss.
    - Called an **arborescence**, found in time *O(|A| + n log n) [Gabov & al., 1986]*.

- But a stage has many events, each with different time gains.



Multiple events in a stage

Have Magnet Weapon

Save 10 secs

Have Rush Jet

Save 20 secs

Multiple events in a stage

- A stage is a set of events, each with different possible gains.

STAGE S_1 =

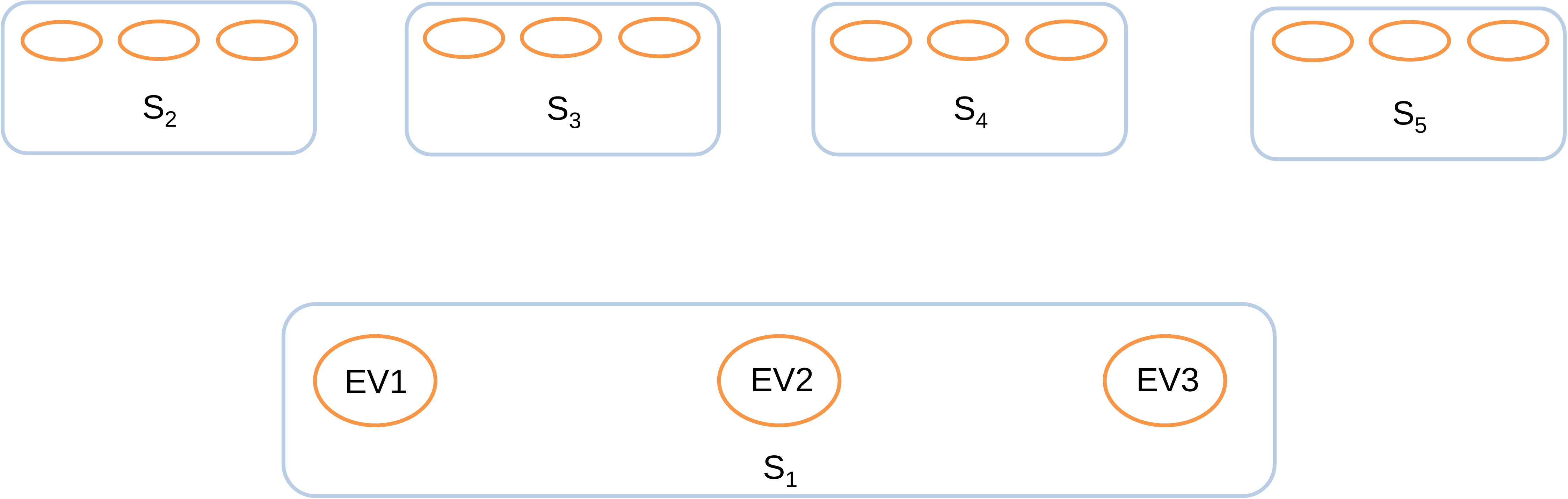
EVENT 1
If S_2 cleared, save 10s
If S_3 cleared, save 5s
If S_4 cleared, save 12s

EVENT 2
If S_3 cleared, save 8s
If S_4 cleared, save 8s
If S_5 cleared, save 2s

EVENT 3
If S_3 cleared, save 4s
If S_5 cleared, save 8s

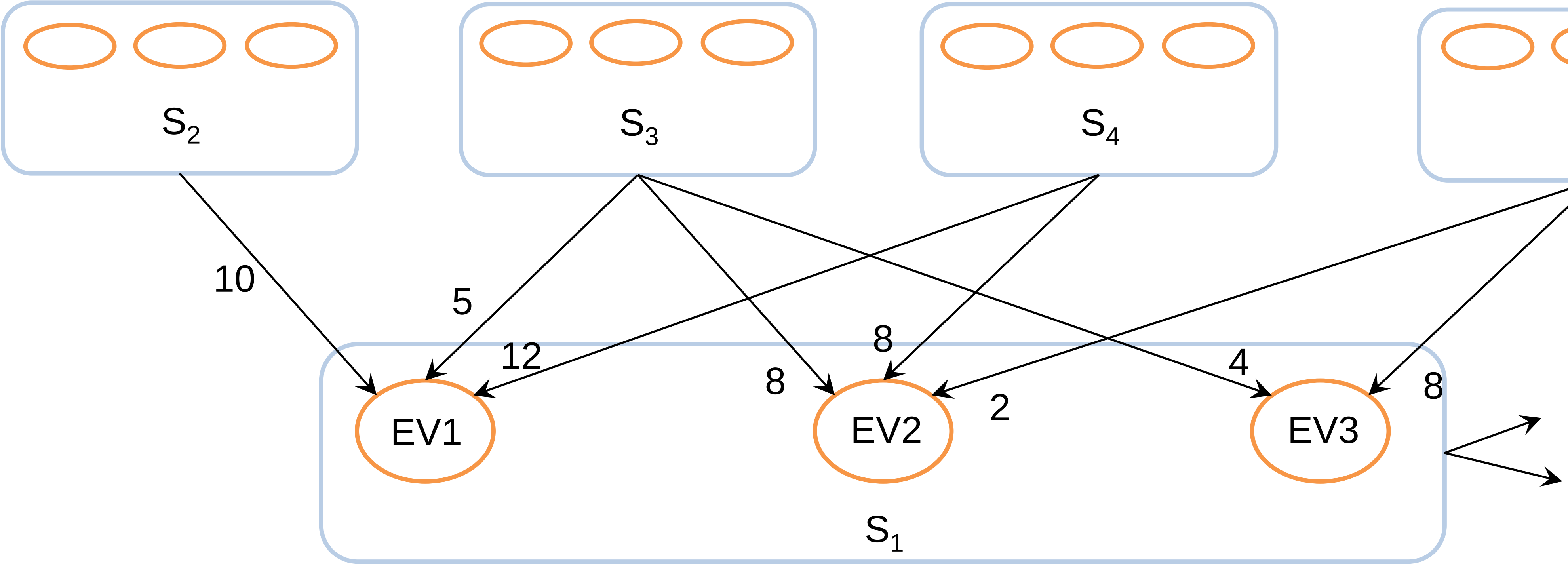
Multiple events in a stage

- A stage is a set of events, each with different possible gains.



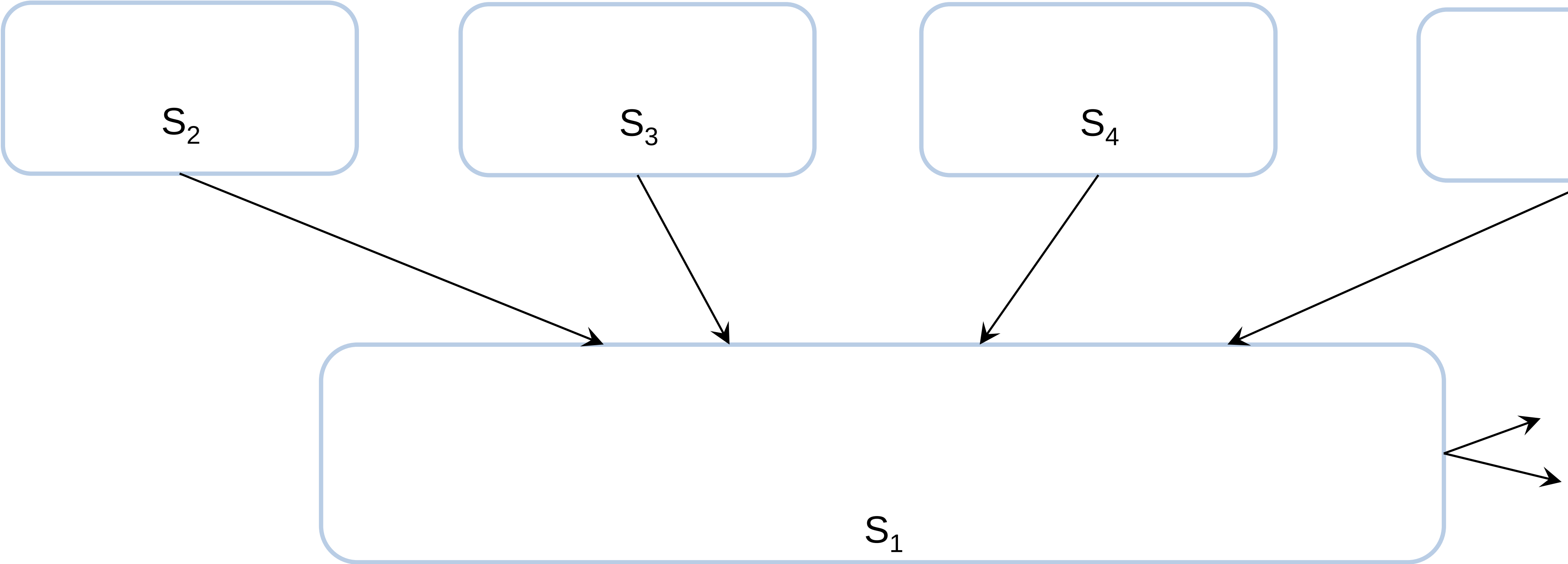
Multiple events in a stage

- A stage is a set of events, each with different possible gains.



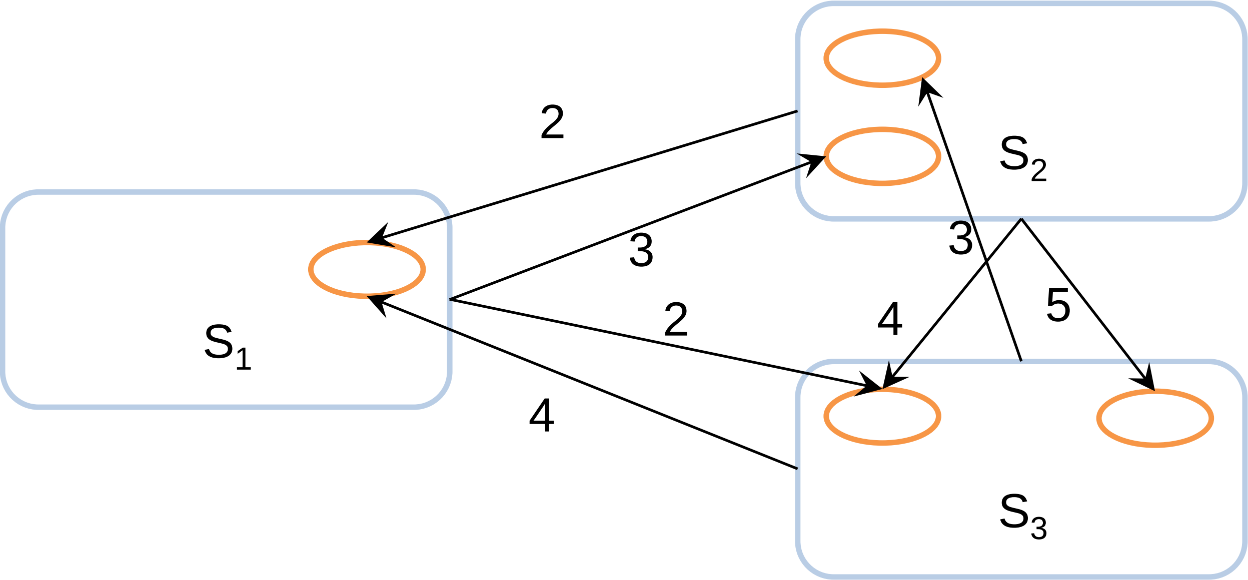
Multiple events in a stage

- **Stage graph** = collapse all events from the same stage.



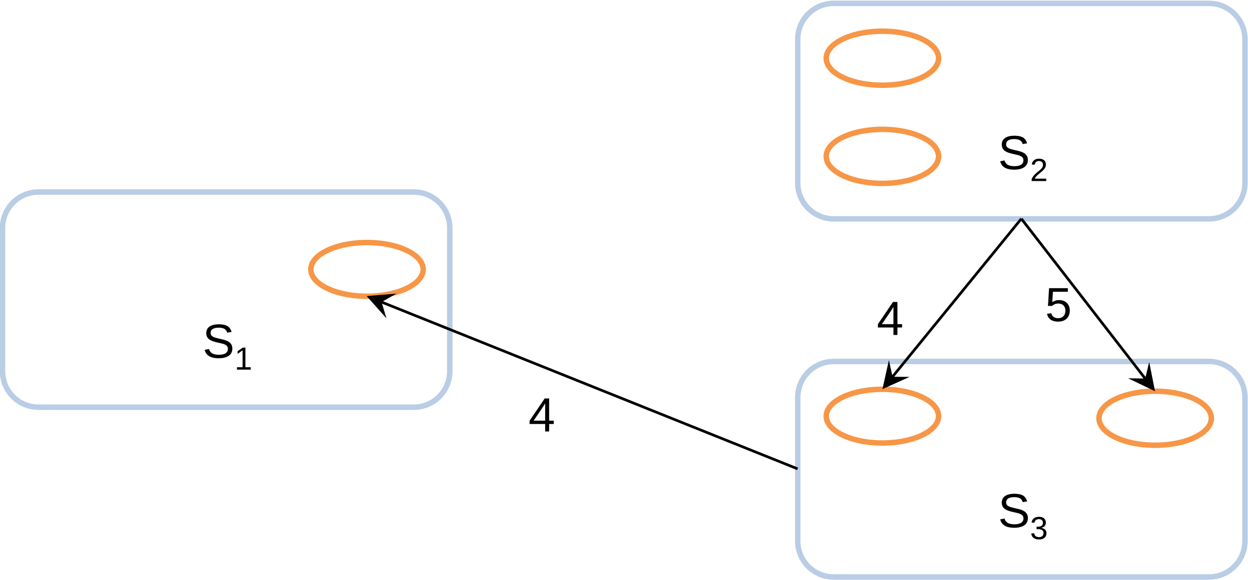
Graph theoretic formulation

- **Given**: a directed graph with *event vertices* of out-degree 0, and *stage vertices* of in-degree 0.
- **Find**: a maximum-weight sub-digraph of in-degree at most 1 such that the stage graph is acyclic.



Graph theoretic formulation

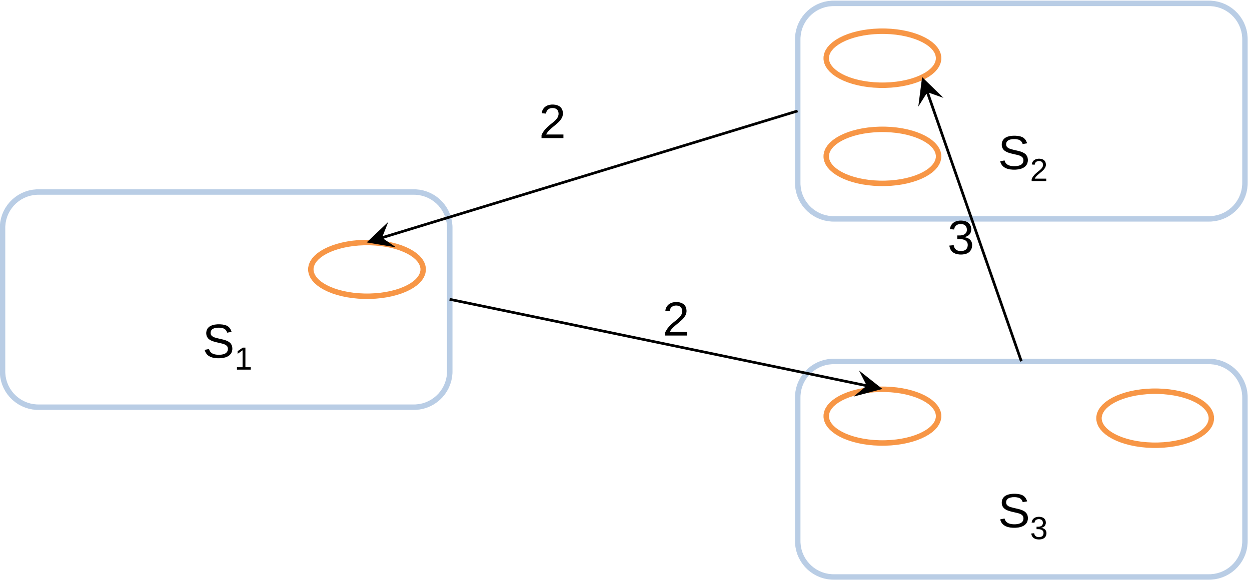
- **Given**: a directed graph with *event vertices* of out-degree 0, and *stage vertices* of in-degree 0.
- **Find**: a maximum-weight sub-digraph of in-degree at most 1 such that the stage graph is acyclic.



GOOD

Graph theoretic formulation

- **Given**: a directed graph with *event vertices* of out-degree 0, and *stage vertices* of in-degree 0.
- **Find**: a maximum-weight sub-digraph of in-degree at most 1 such that the stage graph is acyclic.



BAD

The Stage Routing problem

- **NP-hard** if stages have at most 3 events of in-degree 1 and stages have out-degree at most 2.
  - Follows from results on feedback arc set.

The Stage Routing problem

- **NP-hard** if stages have at most 3 events of in-degree 1 and stages have out-degree at most 2.
  - Follows from results on feedback arc set.
- Approximability
  - **Maximizing time gain** admits a trivial ½-approximation: try any ordering of S. This or its reverse gains time at least ½ OPT.
  - **Minimizing the time gains not taken** is harder:
    - Hard to approximate within a ratio better than $O(\log n)$, even if the **stage graph is a tree** and only one stage has more than 1 event.

The Stage Routing problem

- Fixed-parameter tractability.
  - **W[2]-hard** in the minimum time gains not taken.
  - Cannot be FPT in the **in-degree** or **out-degree** of stage graph.
  - Cannot be FPT in the **treewidth** of the stage graph (both unless P=NP).

The Stage Routing problem

- Fixed-parameter tractability.
  - **W[2]-hard** in the minimum time gains not taken.
  - Cannot be FPT in the **in-degree** or **out-degree** of stage graph.
  - Cannot be FPT in the **treewidth** of the stage graph (both unless P=NP).

  - FPT in $d + t$, where d = maximum in-degree and t = treewidth
    - Use a tree decomposition T with dynamic programming.
    - Main idea: at each bag X of T, try every ordering of $N^-[X]$
    - Simple DP algorithm yields $O((dt)!\ poly(n))$ algorithm.
    - Can be improved to $O(2^{t(d \log d) + d}\ poly(n))$.

Conclusion

- A new framework to treat video games as optimization problems.
- Open problems:
  - FPT status of damage boosting with chicken events.
  - Approximability of damage boosting with lives.
  - Good algorithms for routing stages?
- Other speedrunning mechanics.

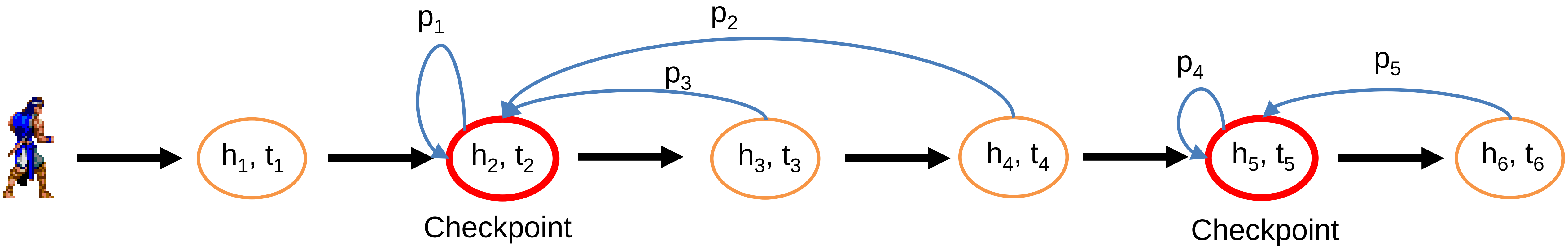
Conclusion

- A new framework to treat video games as optimization problems.
- Open problems:
  - FPT status of damage boosting with chicken events.
  - Approximability of damage boosting with lives.
  - Good algorithms for routing stages?
- Other speedrunning mechanics:
  - Random number generation manipulation
  - Optimizing experience in role-playing games (e.g. Final Fantasy)

Damage boosting with lives

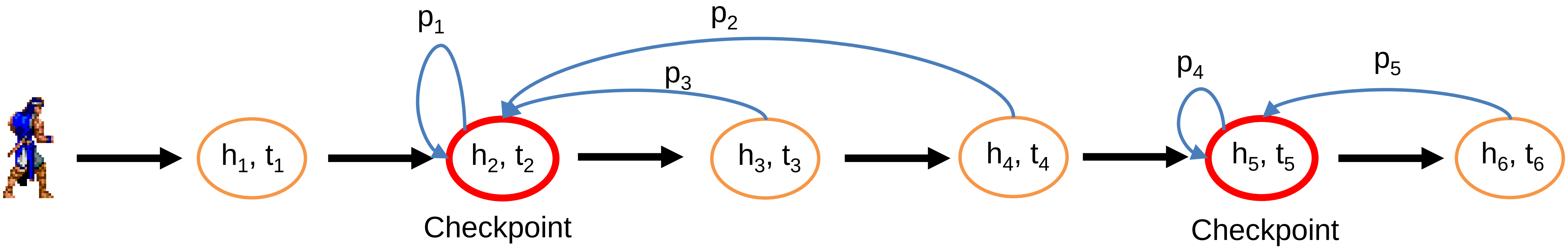
Damage boosting with lives

- Dying also refills health!
  - (it only costs you your life)



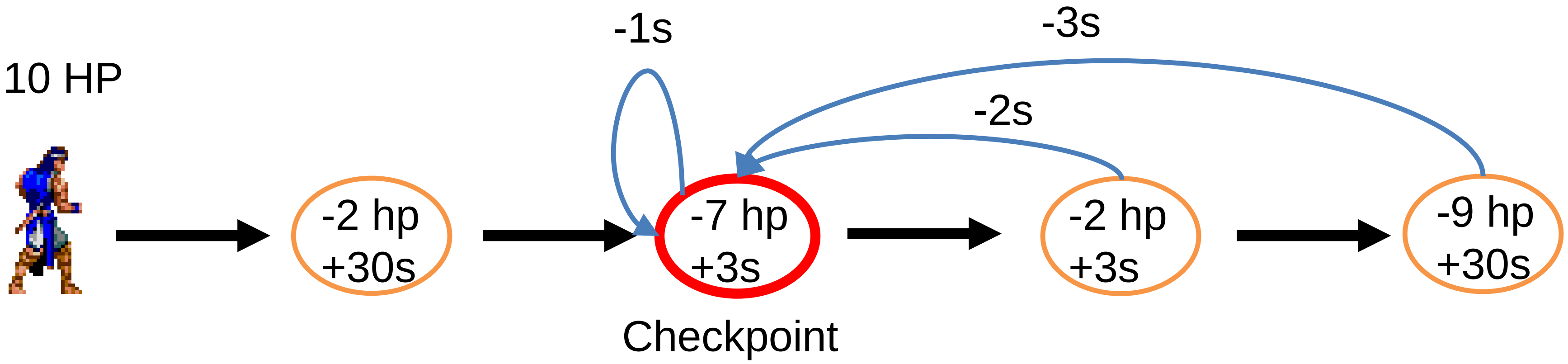
Damage boosting with lives

- Death edges.
  - Can only be taken if hp < 0 after taking event.
  - Restart at last *checkpoint event*.
  - Refill health to 100%.
  - Incur a time penalty p_i.
  - Limited number L of lives (can die at most $L - 1$ times).



Damage boosting with lives

- Death edges.
  - Can only be taken if hp < 0 after taking event.
  - Restart at last *checkpoint event*.
  - Refill health to 100%.
  - Incur a time penalty p_i.
  - Limited number L of lives (can die at most $L - 1$ times).



Damage boosting with lives

- Hard to approximate within a ratio ½, even when $L = 2$.
- If player has L lives, can approximate within a factor $1/L - \varepsilon$.
    - *Trivial algorithm:* do optimal with one life using PTAS.
- Can be solved in pseudo-polynomial time $O(n^2\,(maxHP)^2\,L)$.