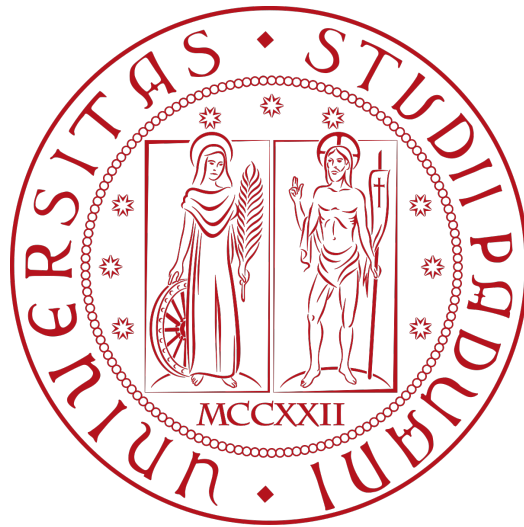


Università degli Studi di Padova



Relazione per: Ricerca Operativa

Per Minimizzare:

il numero di turni per sconfiggere un Boss
nei giochi RPG

Realizzata da:

Gabriele Di Pietro, matricola 2010000

GitHub Repository:

<https://github.com/SerpenTaki/ProgettoR0-2025>

Contents

1	Introduzione	2
1.1	Abstract	2
1.2	Descrizione del problema	2
2	Modello	2
2.1	Insiemi	2
2.2	Parametri	3
2.3	Variabili decisionali	3
2.4	Funzione obiettivo	4
2.4.1	Vincoli	4
3	Implementazione	5
3.1	File .run	5
3.2	File .mod	6
4	Primo scenario e introduzione	8
4.1	File .dat	8
4.2	Risultati	9
5	Secondo scenario	10
5.1	File .dat	10
5.2	Risultati	10
6	Terzo scenario e conclusioni	10
6.1	File .dat	10
6.2	Risultati	10
7	Conclusioni	10

1 Introduzione

1.1 Abstract

Si presenta un problema di ottimizzazione strategica nei giochi RPG dove il giocatore deve comporre un team di personaggi per minimizzare il numero di turni impiegati per sconfiggere un eventuale boss.

1.2 Descrizione del problema

Per un gioco di ruolo il giocatore può scegliere di comporre un team di personaggi. Ogni *party*¹ è formato da 3 personaggi a scelta tra Cavalieri e Maghi. Questo significa che un team può essere formato da 3 Cavalieri, 3 Maghi o da un misto di entrambi. La stamina e il mana sono statistiche separate ma condivise tra i personaggi. In generale il mana viene usato dai Maghi, mentre la stamina dai Cavalieri. I 3 personaggi possono decidere di attaccare durante lo stesso turno o di non attaccare per quello specifico turno per recuperare un certo quantitativo di mana e stamina. I Cavalieri, quando attaccano, possono decidere di usare la spada con 2 mani consumando il 75% di stamina in più ma infliggendo il doppio dei danni. Mentre i Maghi, decidendo di non attaccare, potenziano l'attacco dei Cavalieri del 20% in più per i 2 turni successivi. Non è possibile che 2 Maghi si riposino nello stesso turno. Si vuole minimizzare il numero di turni totali per uccidere un grande mostro con un determinato quantitativo di vita. Sapendo che:

- Un Boss ha un certo numero di punti vita;
- Nei turni di riposo il mago ripristina il 20% di mana consumato;
- Nei turni di riposo il cavaliere ripristina il 30% di stamina consumata;
- Il boost del cavaliere ottenuto dal mago che si riposa aumenta del 20% il danno base con una mano che il cavaliere infligge;

2 Modello

2.1 Insiemi

- T : Insieme dei turni possibili, con $t \in T \subseteq \mathbb{Z}^+$ e nell'implementazione $T = \{1, 2, 3, \dots, 20\}$
- K : Insieme dei cavalieri, con $k \in K \subseteq \mathbb{Z}^+$ e $k = \{0, 1, 2, 3\}$
- W : Insieme dei maghi, con $w \in W \subseteq \mathbb{Z}^+$ e $w = 3 - k$

¹Insieme di cavalieri e maghi che compongono il team.

2.2 Parametri

- PV_{Boss} : punti vita del Boss
- DK : danno che un cavaliere può infliggere
- DW : danno che un mago può infliggere
- SKC : consumo di stamina da parte di un cavaliere
- MWC : consumo di mana da parte di un mago
- S_{max} : stamina massima del team
- M_{max} : mana massimo del team
- S_{min} : stamina minima del team = 0
- M_{min} : mana minima del team = 0

2.3 Variabili decisionali

- S_t : Stamina disponibile all'inizio del turno
- M_t : Mana disponibile all'inizio del turno
- DPT_t : Danno per turno, quindi il danno totale inflitto dal *party* al turno t
- $x_{k,t} = \begin{cases} 1 & \text{se il cavaliere } k \text{ attacca al turno } t \\ 0 & \text{altrimenti} \end{cases}$
- $y_{k,t} = \begin{cases} 1 & \text{se il cavaliere } k \text{ usa 2 mani per attaccare} \\ 0 & \text{altrimenti} \end{cases}$
- $z_{w,t} = \begin{cases} 1 & \text{se il mago } w \text{ attacca nel turno } t \\ 0 & \text{altrimenti} \end{cases}$
- $Boost_t = \begin{cases} 1 & \text{se un mago si è riposato in uno dei 2 turni precedenti} \\ 0 & \text{altrimenti} \end{cases}$
- $u_t = \begin{cases} 1 & \text{se il boss è stato sconfitto nel turno } t \\ 0 & \text{altrimenti} \end{cases}$

2.4 Funzione obiettivo

$$\min \sum_{t \in T} t \cdot u_t \quad (1)$$

L'obiettivo è quello di trovare il numero minimo di turni necessari a sconfiggere il boss.

2.4.1 Vincoli

La stamina e il mana sono valori che si aggiornano in base al consumo (*durante la fase di attacco*) o alla ricarica (*durante la fase di riposo*).

$$S_t \geq S_{min} \quad S_t \leq S_{max} \quad M_t \geq M_{min} \quad M_t \leq M_{max} \quad \forall t$$

$$y_{k,t} \leq x_{k,t} \quad \forall k, t$$

$$\sum_{k \in K} SKC(x_{k,t} + 0.75y_{k,t}) \leq S_t \quad \forall t$$

$$\sum_{w \in W} z_{w,t} \cdot MWC \leq M_t \quad \forall t$$

$$S_{t+1} = S_t - \sum_{k \in K} SKC(x_{k,t} + 0.75y_{k,t}) + 0.3 \cdot r_{k,t} \quad \forall k, t$$

$$M_{t+1} = M_t - \sum_{w \in W} z_{w,t} \cdot MWC + 0.2 \cdot \gamma_{w,t} \quad \forall w, t$$

dove definiamo r e γ come 2 variabili ausiliarie usate per linearizzare il vincolo di riposo dei cavalieri e maghi, e sono soggette a:

$$r_{k,t} \leq S_{max}(1 - x_{k,t}) \quad \gamma_{w,t} \leq M_{max}(1 - z_{w,t})$$

$$r_{k,t} \geq (S_{max} - S_t) - M \cdot x_{k,t} \quad \gamma_{w,t} \geq (M_{max} - M_t) - M \cdot z_{w,t}$$

$$r_{k,t} \leq S_{max} - S_t \quad \gamma_{w,t} \leq M_{max} - M_t$$

$$r_{k,t} \geq 0 \quad \gamma_{w,t} \geq 0$$

Dove M è un numero sufficientemente grande per garantire che il vincolo sia soddisfatto, ad esempio $M = S_{max}$.

Vincoli sul danno e sulla condizione di vittoria

$$Boost_t \leq \sum_{w \in W} ((1 - z_{w,t-1}) + (1 - z_{w,t-2})) \quad \forall t$$

$$z_{w,t-1} + z_{w,t-2} \leq 1 \quad \forall t, w$$

$$DPT_t = (\sum_{k \in K} DK \cdot (x_{k,t} + y_{k,t} + (0.2Boost_t))) + \sum_{w \in W} z_{w,t} \cdot DW \quad \forall t$$

$$\sum_{t=1}^t DPT_t \geq PV_{Boss} \cdot u_t \quad \forall t$$

Vincolo di riposo dei maghi, in un turno solo 1 mago

$$\sum_{w \in W} (1 - z_{w,t}) \leq 1 \quad \forall t$$

Solo un turno può essere quello in cui il boss viene sconfitto

$$\sum_{t \in T} u_t = 1 \quad \forall t$$

Domini:

$$x_{k,t}, y_{k,t}, z_{w,t}, u_t, Boost_t \in \{0, 1\} \text{ in } \mathbb{Z}^+$$

$$S_t, M_t, DPT_t \geq 0, \text{ con } S_t, M_t, DPT_t \in \mathbb{R}^+$$

3 Implementazione

Presento di seguito l'implementazione del modello in AMPL e il relativo file di esecuzione.

3.1 File .run

```
reset;
model file.mod;
data Dati/file.dat;
option solver gurobi;
solve;

display TurniMinimi;
display u, DPT, Stamina, Mana;
```

1: File di esecuzione

3.2 File .mod

```
set T ordered;
set K;
set W;
param PVBoss;
param DK;
param DW;
param SKC;
param MWC;
param Smax;
param Mmax;
param Smin;
param Mmin;
param BigM := Smax;
var Stamina{T} >= 0;
var Mana{T} >= 0;
var DPT{T} >= 0;
var x{k in K, t in T} binary;
var y{k in K, t in T} binary;
var z{w in W, t in T} binary;
var Boost{T} binary;
var u{T} binary;
var r{k in K, t in T} >= 0;
var gamma{w in W, t in T} >= 0;

minimize TurniMinimi:
    sum {t in T} t * u[t];

subject to StaminaMin {t in T: t != last(T)}:
    Stamina[t] >= Smin;

subject to StaminaMax {t in T: t != last(T)}:
    Stamina[t] <= Smax;

subject to ManaMin {t in T: t != last(T)}:
    Mana[t] >= Mmin;

subject to ManaMax {t in T: t != last(T)}:
    Mana[t] <= Mmax;

subject to Attacco2mani{k in K, t in T}:
    y[k,t] <= x[k,t];

subject to ConsumoStamina{t in T}:
```

```

    sum{k in K} SKC * (x[k,t] + (0.75 * y[k,t])) <= Stamina[t
];

subject to ConsumoMana{t in T}:
    sum{w in W} z[w,t] * MWC <= Mana[t];

subject to AggiornaStamina{t in T: ord(t) < card(T)}:
    Stamina[t+1] = Stamina[t] - sum{k in K} SKC * (x[k,t] +
(0.75 * y[k,t])) + 0.3 * sum{k in K} r[k,t];

subject to AggiornaMana{t in T: ord(t) < card(T)}:
    Mana[t+1] = Mana[t] - sum{w in W} z[w,t] * MWC + 0.2 *
sum{w in W} gamma[w,t];

subject to RiposoCavaliere1 {k in K, t in T}:
    r[k,t] <= Smax * (1 - x[k,t]);

subject to RiposoCavaliere2 {k in K, t in T}:
    r[k,t] >= (Smax - Stamina[t]) - BigM * x[k,t];

subject to RiposoCavaliere3 {k in K, t in T}:
    r[k,t] <= Smax - Stamina[t];

subject to RiposoMago1 {w in W, t in T}:
    gamma[w,t] <= Mmax * (1 - z[w,t]);

subject to RiposoMago2 {w in W, t in T}:
    gamma[w,t] >= (Mmax - Mana[t]) - BigM * z[w,t];

subject to RiposoMago3 {w in W, t in T}:
    gamma[w,t] <= Mmax - Mana[t];

subject to BoostCondition {t in T: ord(t) > 2}:
    Boost[t] <= sum {w in W} (1 - z[w,t-1] + 1 - z[w,t-2]);

subject to BoostCondition2 {t in T: ord(t) > 2}:
    sum{w in W}(z[w,t-1]+z[w,t-2]) <= 1;

subject to UnoRiposaPerTurno {t in T}:
    sum {w in W} (1 - z[w,t]) <= 1;

subject to DannoPerTurno {t in T}:
    DPT[t] = sum{k in K} DK * (x[k,t] + y[k,t] + (0.2 * Boost
[t])) + sum{w in W} z[w,t] * DW;

```



```

subject to BossSconfitto {t in T}:
    sum{tt in T: ord(tt) <= ord(t)} DPT[tt] >= PVBoss * u[t];

subject to SoloUnTurno:
    sum{t in T} u[t] = 1;

```

2: Modello in Ampl

4 Primo scenario e introduzione

Negli scenari presentati di seguito, il file `.dat` è stato modificato per variare i parametri del problema, simulando le possibili scelte che un giocatore può effettuare nella composizione del proprio team. Nel primo scenario, il team è composto da 2 cavalieri e 1 mago. Il boss ha 1200 punti vita, e le statistiche di attacco e consumo sono le seguenti:

Parametro	Valore
<i>DK</i>	30
<i>DW</i>	25
<i>SKC</i>	20
<i>MWC</i>	15

Per tutti gli scenari queste statistiche sono state mantenute costanti assieme alla vita del boss. In modo tale che ci si possa concentrare sulla differenza della composizione della *party*.

4.1 File `.dat`

```

set T := 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20;
set K := k1,k2;
set W := w1;
param PVBoss := 1200;
param DK := 30;
param DW := 25;
param SKC := 20;
param MWC := 15;
param Smax := 100;
param Mmax := 100;
param Smin := 0;
param Mmin := 0;
var Stamina := 1 100;
var Mana := 1 100;

```

3: scenario1.dat

4.2 Risultati

L'esecuzione del file `.run` mostra che il numero minimo di turni per sconfiggere il boss è 16. In particolare possiamo notare come appena disponibile il modello predilige l'attacco a 2 mani da parte del cavaliere

```
ampl: include file.run;
Gurobi 11.0.3:                               Gurobi 11.0.3: optimal solution
; objective 16
20975 simplex iterations
624 branching nodes
TurniMinimi = 16

:      u      DPT      Stamina      Mana      :=
1      0      157      91.4286      15
2      0       42      21.4286       0
3      0       37       25          20
4      0      132       70           5
5      0       37        0          24
6      0      102       60           9
7      0       37        5          27.2
8      0      102       62          12.2
9      0       37        7          29.76
10     0      102      62.8         14.76
11     0       37       7.8         31.808
12     0      102      63.12        16.808
13     0       37       8.12        33.4464
14     0       72      63.248        18.4464
15     0       12     39.2736        34.7571
16     1      157     75.7094        47.8057
17     0       12      5.70944       32.8057
18     0        0     62.2838        46.2446
19     0        0     84.9135        56.9956
20     0       12     93.9654        65.5965
;
```

5 Secondo scenario

5.1 File .dat

Per il secondo scenario, il team è composto da 1 cavaliere e 2 maghi. In questo modo si potrà sfruttare al meglio il boost dato dal riposo da parte dei maghi.

```
set K := k1;  
set W := w1,w2;
```

Come precedentemente detto le statistiche del boss e del *party* sono rimaste invariate. Come unica differenza abbiamo quindi la composizione del team di personaggi.

5.2 Risultati

Eseguendo il file `.run` otteniamo che il numero minimo di turni che serve per sconfiggere il boss è 20. E ad ogni turno è stato possibile sfruttare il boost dato dal riposo dei maghi.

6 Terzo scenario e conclusioni

6.1 File .dat

Per il terzo scenario, il team è composto da 3 cavalieri. In questo caso non sarà possibile sfruttare il boost dato dai maghi, tuttavia potremmo attaccare con 2 mani. Questo dovrebbe garantire un quantitativo maggiore di danni e di conseguenza un numero minore di turni.

```
set K := k1,k2,k3;  
set W := ;
```

6.2 Risultati

In questo scenario, il numero minimo di turni necessari per sconfiggere il boss è 15. Questo risultato evidenzia come la composizione del team influenzi significativamente la strategia. Sebbene non sia possibile sfruttare il boost fornito dai maghi, il danno elevato inflitto dai cavalieri, specialmente utilizzando l'attacco a due mani, si dimostra determinante. Di conseguenza, questa strategia risulta più vantaggiosa rispetto a quelle che includono i maghi. Tuttavia, ciò suggerisce che potrebbe essere necessario bilanciare meglio le statistiche dei cavalieri per rendere la scelta dei maghi più competitiva e strategicamente valida.

7 Conclusioni

L'obiettivo del progetto era quello di sviluppare un modello matematico per ottimizzare la composizione del team in un gioco di ruolo, al fine di dimostrare come la Ricerca Operativa possa essere utilizzata a vantaggio dei programmatori per risolvere problemi complessi come il bilanciamento delle statistiche dei personaggi dei videogiochi.