Optimizing Players Operations in the Multiplayer Cooperative Game "Toward the Stars"

F. Liberatore

December 1, 2014

Abstract

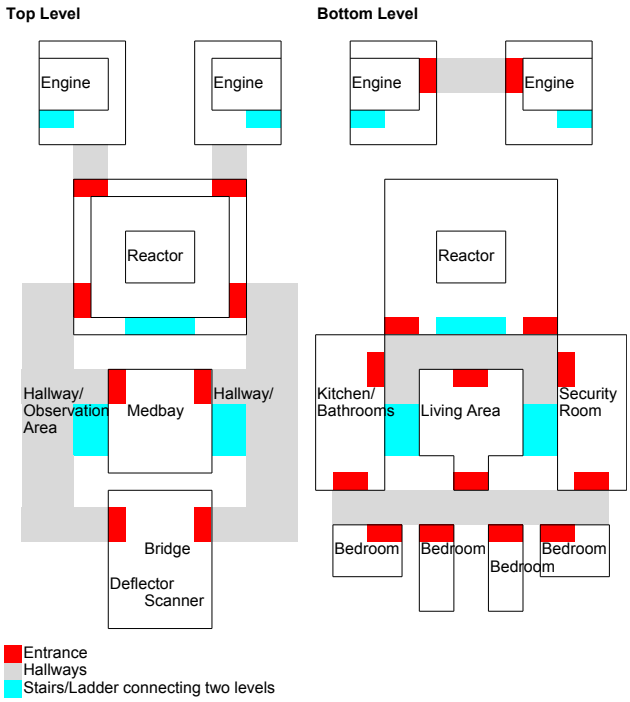
The focus of this project is the optimization of players operations in the multiplayer cooperative game Toward the Stars. The structure of the game is analyzed and an optimization model formulation is proposed. As the model is multi-objective in nature and is comprised of several sub-problems, a specialized decomposition algorithm is proposed. The algorithm is tested on a real-size instance. The computational tests show that the memory and computational requirements of the algorithm are modest. Therefore, the proposed solver could be included as a feature of the videogame in real-time.

1 Introduction

Toward the Stars (TtS) is a cooperative First-Person Action Game with emphasis on spaceship management gameplay. Four players engage in 10-15 minute long missions where they must deal with various events that take place during their starship adventures. The objective of the game is to get to destination and survive all the hazards met during the journey.

This research concerns the definition and optimization of mathematical models that represents the basic mechanics of TtS. Their solutions provide the sequence of operations that the players should take to solve the game in the best possible way. Knowing the best (or a good) solution allows for the extraction of a number of game measures, such as interestingness [1], uncertainty [2], and interaction [3]. These attributes find application in the design phase of the development of the game. In fact, they provide a feedback to game designers regarding many aspects of the game. In the academic literature, automated measurements of games have been used to devise Evolutionary Game Design methods [4], i.e., algorithms capable of designing interesting and challenging games. Similarly, the optimization model for TtS could be the first step toward the definition of a system that automatically defines the game, according to a specified level of challenge.

Figure 1: Spaceship layout.



Additionally, the solution identified by the model can be used as a basis of comparison for the players' actions while the game is running. This, in turn, allows for further analyses, such as assessing their level of skill and cooperation.

2 Structure of the Game

TtS is a complex game that is comprised of numerous features. In this research we focus on the most relevant, so as to obtain a simplified representation of the game that can be modeled mathematically. The game takes place in the players' spaceship. The layout of the spaceship is illustrated in Figure 1. Different types of hazards are possible in the game, such as asteroids, fire, broken equipment, and intruders. Also, the players can use the following tools: laser gun, fire extinguisher, repair tool, and healing spray. Finally, by accessing a special interface the players can use the turrets to shoot down incoming asteroids.

In summary, the game can be segmented into the following basic tasks, each composed of several operations:

- **Getting rid of an intruder**. This task requires the following operations: deciding how many players take part in the task, assigning the available weapons to the participating players, getting the participating players to the assigned weapons locations, getting the armed player to the intruder location, and fighting.

- **Putting out a fire**. This task requires the following operations: deciding how many players take part in the task, assigning the available fire extinguishers to the participating players, getting the participating players to the assigned fire extinguishers locations, getting the equipped player to the fire location, and putting out the fire.
- **Fixing an equipment**. This task requires the following operations: deciding how many players take part in the task, assigning the available repair tool to the participating players, getting the participating players to the assigned repair tools locations, getting the equipped player to the equipment location, and fixing the equipment.
- **Healing a player**. This task requires the following operations: deciding how many players take part in the task, assigning the available healing spray to the participating players, getting the participating players to the assigned healing spray locations, getting the equipped player to the hurt player location, and healing the hurt player.
- **Defending the ship from incoming asteroids**. This task requires the following operations: deciding how many players take part in the task, assigning the available turrets to the participating players, getting the participating players to the turrets locations, and shooting down the asteroids.

It can be easily seen that all the tasks have a very similar structure, that can be expressed as:

1. Deciding how many player take part in the task.
2. Assigning limited equipment to the players.
3. Moving the participating players from their initial location to the assigned equipment locations.
4. Moving the players from the assigned equipment locations to the final location.
5. Specific operations to solve the task.

In the following section, we propose an optimization model for the solution of this generalized task.

3 A Bi-objective Mixed Integer Optimization Model

In this section we propose a Bi-objective Mixed Integer Problem (BMIP) for the solution of a general task in TtS. A MIP is a mathematical model comprised exclusively of:

- Linear equations and inequalities.
- Integer, binary, and continuous variables.

A MIP model is easier to solve than non-linear models and, generally, can be solved directly by means of general optimization packages such as GAMS [5], instead of relying on ad hoc algorithms. Also, modeling a problem provides several insights on its structure that can be exploited to devise faster and more effective methodologies. In the following, the elements comprising the model are presented and explained.

3.1 Sets

$G = (V, A)$ Graph representing the layout of the ship.

V Set of vertices. Indexed by i, j.

A Set of arcs. Indexed by (i, j).

P Set of players. Indexed by p.

E Set of equipment. Indexed by e.

K Number of players. Indexed by k.

3.2 Data

v_p Vertex location of player p.

v_e Vertex location of equipment e.

v_f Final vertex location.

$\mathbf{vcomp}(i, j)$ Function that returns 1 if i is the same vertex as j.

t_{ij} Time taken to go from i to j.

d_i^x Damage taken by a player visiting i, when traveling to the equipment location.

d_i^y Damage taken by a player visiting i, when traveling to the final location.

δ_k Damage taken by k players to complete the task.

τ_k Time required by k players to complete the task.

D Maximum damage sustainable by a player.

T Maximum time to complete the task.

3.3 Variables

n_k Number of players variable. Equals 1 if k players participate to the task.

c_p Task participation variable. Equals 1 if player p participates to the task.

m_{pe} Player to equipment assignment map. Equals 1 if equipment e is assigned to player p.

x_{ij}^p Player p path variable to equipment. Equals 1 if player p traverses arc (i, j) while going to retrieve the equipment.

y_{ij}^p Player p path variable to final location. Equals 1 if player p traverses arc (i, j) while going to the final location.

T^p Total time spent by player p.

W^T Worst (greatest) total time spent.

D^p Total damaged suffered by player p.

W^D Worst (greatest) damaged suffered by a player.

3.4 Constraints

Number of Players Constraints

$$\sum_{k=1}^{K} n_k = 1 \tag{1}$$

$$\sum_{p \in P} c_p = \sum_{k=1}^{K} k n_k \tag{2}$$

$$n_k \in \{0, 1\} \qquad \forall k = 1, \ldots, K \tag{3}$$

Constraint (1) forces to select exclusively one number of players. Constraint (2) defines what number of players is participating in the operations.

Task Participation Constraints

$$\sum_{p \in P} c_p \geq 1 \tag{4}$$

$$c_p = \sum_{e \in E} m_{pe} \qquad \forall p \in P \tag{5}$$

$$|A|\, c_p \geq \sum_{(i,j) \in A} x_{ij}^p \quad \forall p \in P \tag{6}$$

$$|A|\, c_p \geq \sum_{(i,j) \in A} y_{ij}^p \quad \forall p \in P \tag{7}$$

Constraint (4) imposes that at least 1 player must participate in the task. Constraints (5) states that a player that participates must be assigned to an equipment and viceversa. Constraints (6) and (7) link the player participation variables to the path variables.

Player/Equipment Assignment Constraints

$$\sum_{e \in E} m_{pe} \leq 1 \quad \forall p \in P \tag{8}$$

$$\sum_{p \in P} m_{pe} \leq 1 \quad \forall e \in E \tag{9}$$

$$m_{pe} \in \{0,1\} \quad \forall p \in P, \forall e \in E \tag{10}$$

Constraints (8) and (9) impose that a player can be assigned at most to an equipment, and viceversa.

Player-to-Equipment Path Constraints

$$\sum_{j \in V} x_{ji}^{p} + \text{vcomp}\,(i, v_p)\, c_p = \sum_{j \in V} x_{ij}^{p} + \sum_{e \in E} \text{vcomp}\,(i, v_e)\, m_{pe} \quad \forall p \in P, \forall i \in V \tag{11}$$

$$0 \leq x_{ij}^{p} \leq 1 \quad \forall p \in P, \forall\, (i,j) \in A \tag{12}$$

Constraints (11) define the path of a player from his/her initial location to the assigned equipment location.

Equipment-to-Final-Location Path Constraints

$$\sum_{j \in A} y_{ji}^{p} + \sum_{e \in E} \text{vcomp}\,(i, v_e)\, m_{pe} = \sum_{j \in A} y_{ij}^{p} + \text{vcomp}\,(i, v_f)\, c_p \quad \forall p \in P, \forall i \in V \tag{13}$$

$$0 \leq y_{ij}^{p} \leq 1 \quad \forall p \in P, \forall\, (i,j) \in A \tag{14}$$

Constraints (12) define the path of a player from the location of the assigned equipment to the final location.

Attributes

$$T^p = \sum_{(i,j)\in E} t_{ij} x_{ij}^p + \sum_{(i,j)\in E} t_{ij} y_{ij}^p + \sum_{k=1}^{K} \tau_n n_k \quad \forall p \in P \tag{15}$$

$$W^T \geq T^p \quad \forall p \in P \tag{16}$$

$$D^p = \sum_{(i,j)\in E} d_{ij}^x x_{ij}^p + \sum_{(i,j)\in E} d_{ij}^y x_{ij}^p \quad \forall p \in P \tag{17}$$

$$W^D \geq D^P \quad \forall p \in P \tag{18}$$

$$W^D \leq D \tag{19}$$

$$W^T \leq T \tag{20}$$

Constraints (15)-(18) calculate the time taken by each player to complete the task, the total time (computed as the worst time), the damage taken by each player, and the maximum damaged suffered by a player. Constraints (19) and (20) impose that the damage received by a player and the total time cannot exceed D and T, respectively.

Objective Functions

$$\min \quad W^T \tag{21}$$

$$\min \quad W^D \tag{22}$$

The objective of the model is to minimize the total time taken to complete the task and the maximum damage suffered by a player, at the same time.

Discussion

The model is multi-objective in nature and, therefore, it cannot be solved as-is. Also, it is comprised by a large number of binary and continuous variables. This is due to the fact that the model as a whole is comprised by a set of subproblem, as explained in Section 2, and therefore many variables and constraints are required to link and connect its different parts. In the literature on optimization it is a well-known fact that, when possible, it is more efficient to solve many smaller sub-problems rather than a single bigger problem, as the complexity of these problems grows exponentially. In the next section we present an optimization algorithm that exploits the structure of the problem to solve the problem in a very efficient way.

4 A Decomposition Algorithm

As explained in Section 2, the general problem of solving a task is comprised of the following steps:

1. Deciding how many player take part in the task.
2. Assigning limited equipment to the players.
3. Moving the participating players from their initial location to the assigned equipment locations.
4. Moving the players from the assigned equipment locations to the final location.
5. Specific operations to solve the task.

Obviously, every step is interconnected to the others. However, we can solve each subproblem independently and then merge the solutions to find the single best solution to the global problem. In the following, the element comprising the decomposition algorithm are explained.

4.1 Specific operations to solve the task

In terms of the model, the operations necessary to complete the task affect the total time taken to solve the task and the damage suffered by the players. These are given by parameters τ_k and δ_k. Therefore, given a fixed number of players $\bar{k}$, the problem of completing the task is trivially solved by summing $\tau_{\bar{k}}$ and $\delta_{\bar{k}}$, as in Equations (15) and (17).

4.2 Moving the Players to the Equipment and to the Final Location

The objective of this subproblem is to minimize both the time taken to get to destination and the damage sustained by the player along the path. Given $\bar{v}_p$ and $\bar{v}_e$ the locations of a determined player and of a determined equipment, respectively, the resulting problem is a Multiobjective Shortest Path Problem (MSPP) on graph G. The MSPP was originally proposed by Vincke [7] and its objective is to to determine a path that minimizes simultaneously all the criteria under consideration. Usually, there is not a path exhibiting the best value for all the criteria. Hence, for the MSPP, we are looking for the Pareto Frontier (PF), i.e., the set of non-dominated paths in the network, that is, paths for which it is not possible to find a better value for one criterion without getting worse for some of the other criteria. The MSPP is known as being hard to solve and the large number of applications of the MSPP led to the development of various strategies to obtain the optimal solutions.

In this work, we solved the MSPP by using a labeling algorithm that provides PF^{pe} and PF^{ef}, i.e., the Pareto Frontiers comprised of all the paths from $\bar{v}_p$ to $\bar{v}_e$ and from $\bar{v}_e$ to $\bar{v}_f$, that (1) minimize the time

taken and (2) minimize the damage sustained by the player p to get to the equipment e location and to the final location. From PF^{pe} and PF^{ef} we obtain $\bar{PF}^{pe}$ and $\bar{PF}^{ef}$, which are the set of non-dominated paths that are feasible according to Equations (19) and (20). In the following, the MSPP labeling algorithm implemented is presented.

Algorithm 1 Labelling algorithm to solve the MSPP.

```
{Data structures initialization.}
for all i ∈ V do
  Q_i ← ∅; {Initialize a path bin for each vertex as an empty set.}
end for
{Algorithm initialization.}
ρ̄ ← {v_p}; {Creating a path visiting only vertex v_p.}
Q_{v_p} ← {ρ̄}; {Initialize path bin of initial vertex with ρ̄.}
Q̄ ← ∅; {Initialize open paths stack as empty set.}
push(Q̄, ρ̄);
{Loop while there are open paths in the stack.}
while Q̄ not empty do
  ρ̄ ← pop(Q̄); {Get an open path and remove it from the stack.}
  {Skip paths dominated by any path at destination.}
  if ρ̄ not dominated by Q_{v_e} then
    i ← last vertex in ρ̄;
    {Generate paths exiting from the current last vertex.}
    for all j ∈ V s.t. (i, j) ∈ A and j ∉ ρ̄ do
      ρ̄' ← ρ̄ ∪ {j}; {Skip paths dominated by any path at next vertex.}
      if ρ̄' not dominated by Q_j then
        Remove from Q_j all the paths dominated by ρ̄';
        Remove from Q̄ all the paths ending at j dominated by ρ̄';
        push(Q_j, ρ̄'); {Push ρ̄' into the corresponding bin Q_j.}
        {Check if last vertex is not the destination node.}
        if j ≠ v̄_e then
          push(Q̄, ρ̄'); {Push ρ̄' into the open paths stack.}
        end if
      end if
    end for
  end if
end while
return Q_{v_e}; {Return the non-dominated paths at destination.}
```

The algorithm finds PF^{pe}, the Pareto Frontiers comprised of all the paths from $\bar{v}_p$ to $\bar{v}_e$, and it can be easily modified to compute PF^{pf}. Initially, the algorithm creates an empty set for each vertex $i \in V$, an empty stack of open paths, and an initial path starting at $\bar{v}_p$, which is then pushed into the stack of open paths and placed into the path set corresponding to $\bar{v}_p$, $Q_{\bar{v}_p}$. After the initialization, the algorithm runs the main loop until the stack of open paths is empty. In the main loop, a path is popped from the open paths stack. Then, a bounding criteria is applied: the current path is skipped if it is dominated by any path in the destination vertex set $Q_{\bar{v}_e}$. The dominance rule is the following.

$$\rho_A \text{ dominates } \rho_B \; iff \tag{23}$$

$$time\,(\rho_A) \leq time\,(\rho_B) \tag{24}$$

$$damage\,(\rho_A) \leq damage\,(\rho_B) \tag{25}$$

$$\text{At least one inequality is strict.} \tag{26}$$

where $time\,(\rho)$ and $damage\,(\rho)$ are the values of the time and the damage attributes, respectively.

If no path at destination dominates the current path, then children paths are generated by adding to the current path all the vertices connected to its last vertex, one for every children. Each children path is added to the path set corresponding to its last node. This operation involves checking for dominance between the children path and the paths in the set. All the paths that are found dominated are removed from the set and from the open path stack when necessary. Finally, if a child path is not dominated and it does not reach $\bar{v}_e$, it is added to the stack of open paths.

4.3 Assigning the Equipment to the Players

Once we solve the path subproblems, we can compute $\bar{PF}^{pef}$, the Pareto Frontier of the feasible paths connecting a player p to the final location v_f when assigned to the equipment$e$:

$$\bar{PF}^{pef} = \left\{ \rho_{pef} = \{\rho_{pe} \cup \rho_{ef}\} \middle| \rho_{pe} \in \bar{PF}^{pe}, \rho_{ef} \in \bar{PF}^{ef}, time\,(\rho_{pef}) \leq T, damage\,(\rho_{pef}) \leq D \right\} \tag{27}$$

Let $q = 1, \ldots, \left|\bar{PF}^{pef}\right|$ be the index of a path ρ^q_{pef} inside $\bar{PF}^{pef}$. Once we have all $\bar{PF}^{pef}$ we can compute the best way of assigning players to fire extinguishers. Now, let $\bar{k}$ be the fixed number of players participating in the task and w^q_{pe} be binary variables taking value 1 if the player p is assigned to the fire extinguisher e and follows path ρ^q_{pef}, and 0 otherwise. The problem of determining the best assignment of fire extinguishers to

players can be formulated as:

$$\min \quad \max \ time\left(\rho^q_{pef}\right) w^q_{pe} \tag{28}$$

$$\min \quad \max \ damage\left(\rho^q_{pef}\right) w^q_{pe} \tag{29}$$

$$s.t. \quad \sum_{e\in E}\sum_q w^q_{pe} \leq 1 \qquad \forall p \in P \tag{30}$$

$$\sum_{p\in P}\sum_q w^q_{pe} \leq 1 \qquad \forall e \in E \tag{31}$$

$$\sum_{p\in P}\sum_{e\in E}\sum_q w^q_{pe} = \bar{k} \tag{32}$$

$$w^q_{pe} \in \{0,1\} \tag{33}$$

The objective is to minimize the worst time and the worst damage. Constraints (30) and (31) impose that a player can have at most one equipment assigned, and that an equipment can be assigned to one player at most. Finally, constraint (32) imposes that the number of participating players must be exactly $\bar{k}$. This problem is a generalized assignment problem with min-max objective function [8].

For its solution we propose a bounded enumeration algorithm that solves the problem for all the values of $\bar{k} = 1, \ldots, K$ at the same time, since the solution space of the problem for a lower value of $\bar{k}$ is entirely contained in the solution space of the problem for a higher value of $\bar{k}$. The algorithm features a bound that excludes partial solutions that cannot lead to any optimal solution, reducing the computational complexity of the problem significantly. The algorithm computes the sets W^k of efficient assignments with k players.

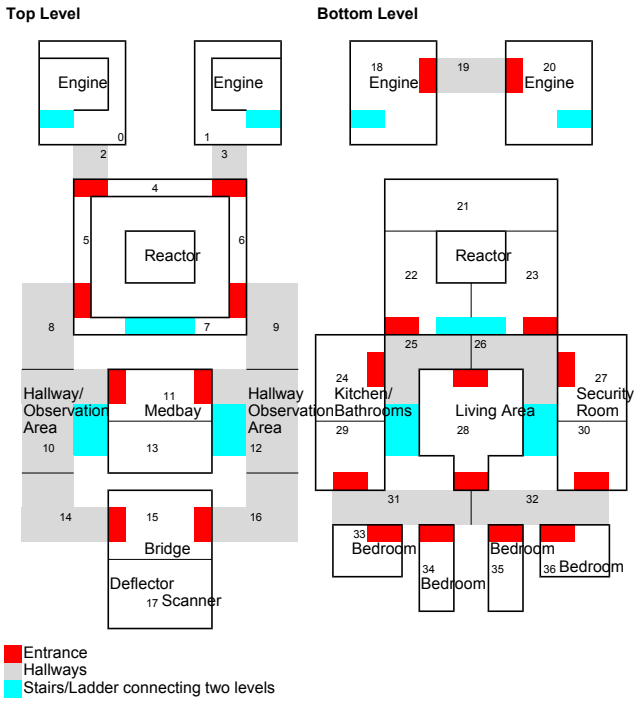
5 Computational Experience

5.1 Dataset

The solver is run on a graph obtained by subdividing the rooms of the ship into spaces having the same size, approximately. The final configuration is shown in Figure 2. Two arcs are created for each pair of neighboring areas. All the arcs have unitary length. The resulting graph is comprised of 37 nodes and 102 arcs. Other relevant data is given in the following.

- Initial players locations (v_p): nodes 0, 15, 24, and 36.
- Equipment locations (v_e): nodes 20, 24, and 27.
- Event location (v_f): node 18.
- Damage taken by the players to complete the task (δ_k): 0 for all the cases.

Figure 2: Spaceship layout.



- Time required by the players to complete the task (τ_k): $\{6, 4, 2, 1\}$.
- Maximum damage sustainable by a player (D): 2.
- Maximum time to complete the task (T): 100.
- The damage associated to the arcs is zero, except for the arcs entering into node 19 (the event location) that have an associated damage of 1.

5.2 Results

The algorithms have been programmed in C++. The experiments have been run on a computer equipped with an Intel Core i7-4500U CPU and 4GB RAM memory, using only one core per run.

Memory and Computational Time Requirements The algorithm requires 360KB of physical memory.To test the computational requirements, we run the algorithm 10,000 times on a single core. The whole run took only 61.3594 seconds and, therefore, the average running time of the algorithm is 0.00613594 seconds, with the considered parameters. Given the limited memory and computational requirements, the algorithm can be used in real-time while the game is running.

Table 1: Solution using only one player. Time = 13, Damage = 1.

Player	Tool	Time	Damage	Route
1	1	13	1	0, 2, 4, 3, 1, 20, 19, 18

Table 2: Solution using two players. Time = 12, Damage = 1.

Player	Tool	Time	Damage	Route
1	1	11	1	0, 2, 4, 3, 1, 20, 19, 18
3	2	12	1	24, 25, 22, 7, 6, 4, 2, 0, 18

Sample Solutions The algorithm identifies one solution using only one player (Table 1), one solution using two players (Table 2), and two solutions using three players (Table 3). The solutions using three players are dominated by the others. In fact, the solutions using one and two players have the same objective function values (or better) and a lower number of players than those obtained using three players. Therefore, we can disregard the solutions obtained using three players.

6 Conclusions

In this work, an optimization model for the definition of players operations in the face of hazardous events in the game Toward the Stars has been presented. The model is solved by exploiting its structure. Specifically, the problem is decomposed into subproblems that are then solved by means of specialized algorithms. As the global problem is multi-objective in nature, multiple optimal solution are obtained at every step of the process. The algorithm is tested on a graph representing the current ship layout. The experiments show that the algorithm requires 360KB and that a single run of the solver takes only 0.00613594 seconds on average. Therefore, the algorithm can be used in real-time as part of the game Artificial Intelligence engine.

The next step in the research will be the development of an algorithm capable to solve the problem of

Table 3: Solutions using three players.

(a) Time = 13, Damage = 1.

Player	Tool	Time	Damage	Route
1	1	9	1	0, 2, 4, 3, 1, 20, 19, 18
3	2	10	1	24, 25, 22, 7, 6, 4, 2, 0, 18
4	3	13	1	36, 32, 30, 27, 26, 23, 7, 6, 4, 2, 0, 18

(b) Time = 13, Damage = 1.

Player	Tool	Time	Damage	Route
2	1	12	1	15, 16, 12, 9, 6, 4, 3, 1, 20, 19, 18
3	2	10	1	24, 25, 22, 7, 6, 4, 2, 0, 18
4	3	13	1	36, 32, 30, 27, 26, 23, 7, 6, 4, 2, 0, 18

assigning the players to multiple tasks at the same time. A first approach might rely on the iterative solution of the tasks, according to a pre-determined ranking order of the tasks.

References

[1] I. Althöfer, Computer-aided game inventing, Technical Report, Friedrich Schiller Universität Jena, 2003 [Online].

[2] H. Iida, K. Takahara, J. Nagashima, Y. Kajihara and T. Hashimoto, An application of game-refinement theory to Mah Jong, ICEC 2004 (LNCS 3116), M. Rauterberg Ed. Berlin: Springer, 445-450, 2004.

[3] C. Browne, Connection Games: Variations on a Theme, Natick, Massachussetts: AK Peters, 2005.

[4] C. Browne and F. Maire, Evolutionary Game Design, IEEE Transactions on Computational Intelligence and AI in Games 2(1):1-16, 2010.

[5] General Algebraic Modeling System (GAMS), GAMS Development Corporation, http://www.gams.com.

[6] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1:269-271, 1959.

[7] P. Vincke, Problèmes multicritères, Cahiers du Centre d'Etudes de Recherche Opérationelle, 16, 425-439, 1974.

[8] S. Martello, P. Toth, Generalized assignment problems, in Algorithms and Computation, Lecture Notes in Computer Science, 650:351-369, 1992.