Thomas Tregel, Johannes Alef, Stefan Göbel, Ralf Steinmetz: Towards Multiplayer Content Online Adaptation using Player Roles and their Interactions. In: Dr Maja Pivec, Dr Josef Gründler: Proceedings of The 11th European Conference on Games Based Learning, p. 677 - 686, Academic Conferences and Publishing International Limited, October 2017. ISBN 978-1-911218-56-2.

Towards Multiplayer Content Online Adaptation using Player Roles and their Interactions

Thomas Tregel, Johannes Alef, Stefan Göbel, Ralf Steinmetz
TU Darmstadt, Darmstadt, Germany
thomas.tregel@kom.tu-darmstadt.de
johannes.alef@kom.tu-darmstadt.de
stefan.goebel@kom.tu-darmstadt.de
ralf.steinmetz@kom.tu-darmstadt.de

Abstract: Many multiplayer games feature distinct classes or roles for the players to choose from. This choice however, is often limited, as specific group compositions are required to overcome given individual challenges. This limitation can be to the detriment of player enjoyment if it prevents them from playing their favourite role. To cater every challenge for every possible group composition is infeasible due to the high amount of possible groups and their respective role interactions. These aspects are particularly relevant in the context of game-based learning, when adapting content to the current group of players.

To offer a feasible solution to the problem, this paper examines how player roles and their interactions in relation to game content can be modelled and used in an automatic adaptation algorithm. Properties of different roles and their interactions are integrated as group properties. Parameters representing adaptable parts of the game's content are used to model tasks for the group. A suitable balance between the group's properties and the tasks is achieved when the group can overcome the tasks but is still challenged by them. By modelling the balance between the group's properties and the game content as an optimization problem, it allows for fast and efficient determination of an adaptation for the content that offers a suitable balance for any given group. As interactions between roles rarely change between different content sections the modelling effort for groups per section is relatively low. Additionally, sections mostly use game content with comparable game mechanics. Therefore the largest part of the model only has to be determined once and needs to be modified for small model additions.

This paper presents an adaptation algorithm employing a mathematical formalization of role interactions and game content. The algorithm has been implemented in MATLAB and evaluated with three selected optimization algorithms. However, the algorithm's concept is independent of any specific optimization algorithm.

Keywords / Key Phrases: content adaptation, multiplayer, group modelling, mathematical optimization

1. Introduction

Many games offer their players many different roles to choose from. Especially cooperative games use roles, as these foster cooperation between the players. Massively Multiplayer Online Role-Playing Games (MMORPGs) often rely on complex interactions between different roles to create challenging content for the players. The most common roles in these games are Tanks, Damage Dealers and Healers. Tanks protect the other players and take most of the hits, Damage Dealers try to kill the enemies and Healers ensure that everyone survives. A different example could be a sports game, where players take different positions, such as attacker, defender or goalie.

The common aspect of these games is that players take on roles with different responsibilities and the game offers them challenges that they have to overcome by fulfilling their responsibilities. These challenges are often designed for specific group set-ups. Groups with a different distribution of roles have little to no chance of overcoming these challenges in the majority of cases. Sometimes games may even simply impose different roles directly on the players by requiring all roles to be chosen by a specific number of players.

Requiring specific group compositions to play a game successfully can be a problem, as it might force players to choose a role they do not want to play but that is needed to win the game. This might reduce the enjoyment for these players while playing the game, which might in turn keep players from playing the game.

As manually catering each challenge to every possible group would be infeasible, in this paper we present an adaptation model that can be used to automatically adapt games and their challenges in a

way that allows for any composition of roles in groups to overcome challenges while keeping a given metric within a desired range. One example for such a metric would be the difficulty of the challenge, i.e. no matter how the group is composed the difficulty remains the same.

To this end several problems must be solved. First, an adaptation model for cooperative role-based games requires that groups of players can be modelled with respect to their roles and the interactions between the different roles. This model will be derived from an analysis of existing cooperative role-based games. Also, there are different scenarios the adaptation model could be used in that must be considered for the technical requirements of the adaptation model. This requires the consideration of different approaches to the problem and their properties.

2. Related Work

Adaptation in games is concerned with changing aspects of a game to achieve a specified value in a given metric. The changes can pertain to game settings, mechanics or content of the game. In most cases the adaptation uses information about the player for the metric. There are several different goals that can be achieved by the means of adaptation and hence also several different metrics.
Especially in singleplayer games online adaptation is often used to adapt the current game's content to the specific user as seen for the Super Mario series (Pedersen et al. 2009, 2010, Togelius et al, 2011) even going into aspects of emotional research. Missura and Gärtner (2009) argue that dynamic difficulty adjustment is better than a static difficulty level chosen by the player, since the player would need to choose from many different options to get an optimal difficulty level. However choosing the right one from so many options might already prove to be futile and time consuming. Since an optimal difficulty level is important for player satisfaction, providing a dynamic difficulty adjustment that pertains to the abilities of the player is very helpful to games.

As proposed by Yannakakis and Togelius (2011) for procedural content generation different types of evaluation functions can be used to incorporate the user's current status into the ongoing game.
Many approaches used for content adaptation or procedural content generation are optimization problems, where the goal is to find the minimum or maximum for a specific measure, the objective function, for the given problem. As described by J.W. Chinneck (2006) there is a plethora of algorithms to solve the different types of optimization problems. One distinction that can be made for these algorithms is whether they are purely analytical or they employ heuristic elements.
For the latter case two prominent approaches used for our approach are genetic algorithms based upon the principles of evolution (D. Whitley, 1994) and the metaheuristic search method simulated annealing (Kirkpatrick et al., 1983), which is inspired by the annealing process in metallurgy.

For group balancing approaches Konert et al (2013) has shown a suitable approach for multiplayer settings by matching the current player base to the given content. In multiplayer content generation and adaptation, it is important to maintain a level of interaction by personalizing the content (Tregel et al., 2016).

3. Problem Analysis

While many games offer a few unique roles for their players to fulfil, there are some role archetypes that can be found throughout many different games and genres. Most of them are tied to the so called *Holy Trinity* that refers to the three archetypes *Tank, Healer* and *Damage Dealer*. In a qualitative role analysis we looked at multiple leading representatives for the genres of Massively Multiplayer Online Role-Playing Games (MMORPGs), Team-based First Person Shooters and Multiplayer Online Battle Arena (MOBA) to identify commonly used archetypes, roles and subroles.

Roles within these games are identified and distinguished by the tasks they perform well and badly at. What makes them good or bad at these tasks are their different properties or features. Thus, roles can also be identified without considering tasks, by analysing their properties with the addition of interactions between these properties, based on the role distribution present in the group.
Interactions between properties are important to consider when determining which tasks a group can tackle, since cooperation can change the tasks' overall difficulty. Properties can have three basic forms of interaction, where the first one describes the state where no direct influence is present and the second and third one describe the form of positive respectively negative effect on other properties, the synergy and antergy effects.

However different roles are only necessary when there are different tasks that need to be solved and when there are no roles that excel at all possible tasks. The other way around, if two groups differ in their role composition, they can take on different tasks with different intensities.
As for the role and group properties, there are many possible tasks in games, which can be divided along two dimensions: the task's target and its type. The commonly found targets for tasks are enemies, allies and the environment, where an example for the latter is a task that resolves around crossing dangerous areas within a given time period. Regarding the task type tasks can be divided into active and passive tasks, where active tasks require specific actions or action sequences and passive tasks have a strong connection with the player's passive properties like for example his virtual health pool.

4. Concept

To design an adaptation model that pertains to the roles the players have chosen, a mathematical formalization of roles and tasks is necessary. Thus, in this section, mathematical formalizations of the different aspects related to roles will be defined.

4.1 Mathematical Model – basic entities

A task is an atomic challenge where its magnitude or extent is represented by a real number, where higher values indicate a higher challenge or difficulty of the given task.
A property of a role or group is an attribute that can take on different real values, denoting how strong this property is on the respective role or group. An example for such a property is the maximum health of a player.
A role is a collection of extents of all properties
A group is an integer valued vector, where each entry specifies how many players with that specific role are present in the current group.
A role may be computed from a set of attribute values and abilities in conjunction with a mapping of influences of attributes on abilities and a function to calculate the role properties of these inputs.

4.2 Connecting the basic entities: balance function / optimization function

In this section the balance function that is used as the model's objection function in the adaptation will be derived.

Each task is associated with a subset of group properties. This relationship is modelled as a function that returns a value describing how well the group properties are suited for the given task. A zero return value indicates a perfect challenge of this task. Positive values indicate that the challenge of the task is too high for the group, while negative values indicate that the task will note pose much of a challenge. This relationship will be called the balance between task and properties

By calculating the balance between the tasks and the role properties the model is generic and thereby useable for comparable application scenarios. For scenarios without predetermined roles each player's properties can be treated as a different role and are thereby useable in the given model.

For a small example assume that we have a player with the attributes *Health* with the value of 100 and *Healing Power* with the value of 50, as well as two abilities to heal damage suffered from enemies: *Flash Heal* and *Greater Heal*. *Flash Heal* does a considerable amount of healing, while *Greater Heal* inflicts double the amount of healing. A third ability called *Rejuvenation* cannot be performed by our player and does half the healing of *Flash Heal*. The player's role is now modelled as follows:

$$attributes = (Health \quad HealingPower) = (100.0 \quad 50.0)$$
$$abilities = (FlashHeal \quad GreaterHeal \quad Rejuvenation) = (1.0 \quad 1.0 \quad 0.0)$$

The matrix mapping attributes on abilities can be modelled as:

$$attribute\ ability\ map = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 1.0 & 2.0 & 0.5 \end{pmatrix},$$

where each column stands for one of the three abilities, the rows for the effect of Health and *Healing Power* on these abilities. Now if attributes, *attribute ability map* and abilities are multiplied we get:

$(1.0 \quad 1.0 \quad 0.0) * \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 1.0 & 2.0 & 0.5 \end{pmatrix} * \begin{pmatrix} 100.0 \\ 50.0 \end{pmatrix} = 150.0$ which is the effective healing the player can do. The role properties are $\begin{pmatrix} 100 \\ 150 \end{pmatrix}$ as the *Health* is not affected by the *Healing Power* attributes.

In order to calculate the balance we assume an enemy can deal a certain amount of damage to the players. The players can use healing abilities to heal the inflicted damage. For simplification, only the healing properties are considered. A set of roles with two additional roles is given as $(150.0 \quad 50.0 \quad 0.0)$ with one role which can heal a moderate amount of damage and one that cannot heal at all. A group of three players will be given by their role quantity as $(2 \quad 0 \quad 1)$.

The group's effective healing is calculated as $(2 \quad 0 \quad 1) * \begin{pmatrix} 150.0 \\ 50.0 \\ 0.0 \end{pmatrix} = 300.0$

A task called *Enemy Damage* which consists of the enemy's attack damage and its attack rate is calculated by:

$TaskExt(AttackDamage \quad AttackRate) = AttackRate * AttackDamage = 1.5 * 150.0 = 225.0.$

The relationship between the group's effective healing and the enemy's effective damage is linear. For every point of damage inflicted, one point of health must be restored, leading to the balance function:

$b = \|balance(TaskExt, GroupExt)\|_2 = \|TaskExt - GroupExt\|_2 = \|225.0 - 300\|_2 = 75.0.$

For this example the group is not challenged appropriately, because the enemy's damage is too low. Multiple results are possible for the adaptation with the two simplest ones being: The *Attack Damage* is changed to 200.0 or the *Attack Rate* is changed to 2.0. With one of these changes the balance function would return 0.0 representing a perfect balance between task and group properties.

4.3 Towards the optimization model

While the function describes the balance between group properties and tasks perfectly, it may not be suited well for many optimization algorithms. While linear balance function will pose no problem and therefore need no specific handling, nonlinear function might become difficult to handle depending on their properties.

Smooth function, meaning functions that are continuous and whose first derivative is also continuous are manageable by most algorithms, as they allow for using gradient-based approaches to find optima.

Some tasks may have nonlinear non-smooth relationships with group properties and not be suited for approximation. These tasks can be remodeled as constraints. One example for such tasks would be a task that compares a group property to a maximum value in an inequality way. This might for example be the case when there are several sources of damage that can be adapted individually, but the maximum amount of damage any of these can do to a player with a single attack should still be lower than the player's health. One might model this as one inequality, simply using the maximum function. This function is not smooth and therefore not suited for the adaptation. The task can be split up into several inequality constraints, one for each possible value that would otherwise be integrated into the maximum. Thereby the task's effects would not change, but are better suited for optimization, as every constraint would be linear and therefore also smooth.

Because the balance function might return negative balance values a minimization approach would be impractical for the optimization. However, the optimization goal is to reach close to perfect balance. Thereby we apply the Euclidean norm to the resulting vector of balance values.

To keep a level's feeling distinct and to prevent unreasonable adaptation results, bounds for parameters and tasks are employed. They can be used to prevent too low or too high values for the parameters and tasks and thus ensure that any solution found will be playable.

Additionally designers can use these bounds to ensure that the level keeps its identity. These bounds will be used in the form of minimum and maximum values. They can also be used to set certain parameters to specific values if upper and lower bounds are equal. This might be useful if a parameter may be dependent on previous level adaptations, e.g. an enemy that is generally adaptable and appears in multiple levels but should not change between levels.

The overall optimization workflow is depicted in Figure 1.

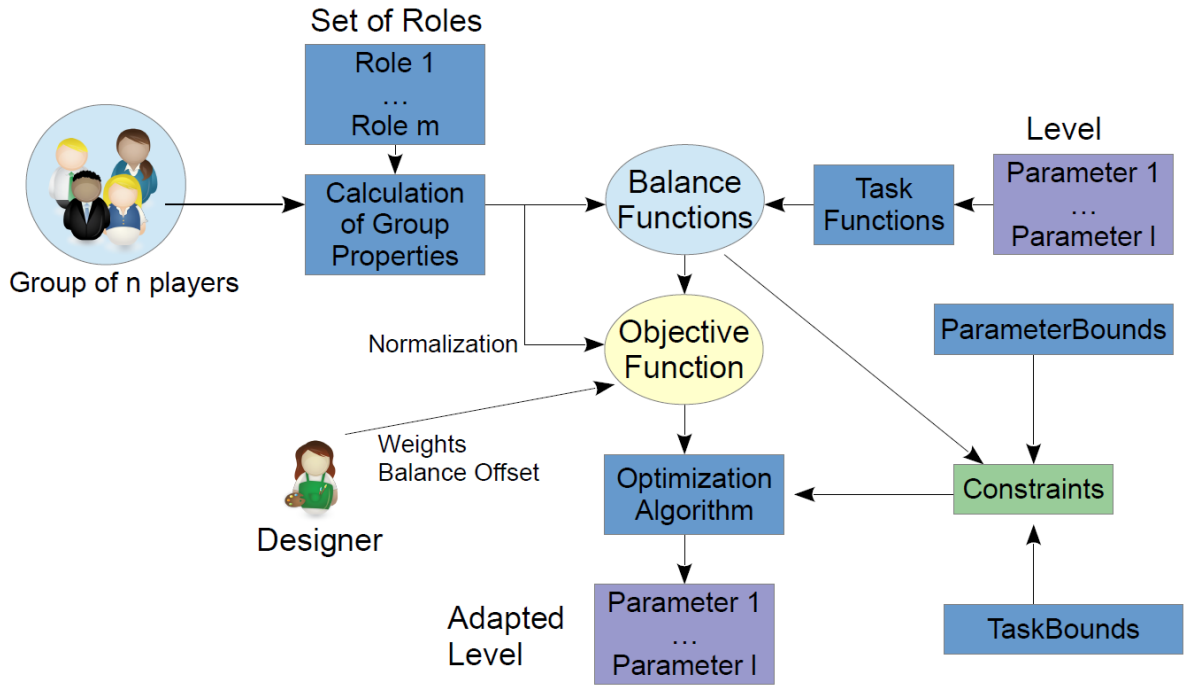

Figure 1: Optimization workflow for content adaptation.

4.4 Transition Model

Sometimes it may be easier to split a level into several small parts and optimize all of them separately. This may for example be the case if developers are trying to reduce the complexity of interactions between mechanics, because there are significant differences in the tasks or to reduce the waiting time for the first part of the level.

In the case this is done two factors must be considered. First, changes to the group or role properties from one segment to the next and second, parameters present in several phases. The first case may be based on the use of limited resources or the acquisition of new items, for example. The second case may be based on enemies present in several phases, for example.

For the transition of parameters and tasks, if a parameter or task has been present in one of the previous optimizations and should not be allowed to change its value, the last value can be set as lower and as upper bound for the task or parameter respectively. That way they will not be changed in the next segment, rather other values will be changed.

Employing such a transition model for large systems, a significant decrease in the overall time used for the adaptation can be expected, as many smaller models for optimizations are often solvable in less time than one large model.

5. Evaluation

The optimization problem described in the previous section can be solved with various different algorithms. In this Section, several different algorithms are tested for their applicability on the model. To this end, example problems for our adaptation model are randomly generated in different sizes and the different algorithms are used to solve these problems. Afterwards they are compared regarding their execution time and the quality of the solution they found.

The evaluation is done in MATLAB (Version 2015b), employing (Global) Optimization Toolbox 7.2. This toolbox provides a wide array of solvers for different types of optimization problems. Since MATLAB is professional software which is widely used, the algorithms can be assumed to be efficiently implemented and the evaluation results therefore not affected by implementation issues.

5.1 Test case setup

There are four main parameters to the adaptation problem: the number of Group Properties, the number of Balance Functions used in the objective function and in the constraints, and the number of parameters.

Four different sizes for the problem are tested. Each set consists of ten randomly generated problems with the same properties as the adaptation model. The results are averaged over the examples within each set. By using multiple problems, exceptionally good or bad results of an algorithm for one specific problem will have less influence. The sets used for testing are given in Table 5.1. It starts with smaller problems that may be used in games with only a few simple role-based mechanics, and ranges to problem sizes that may be found in highly complex games.

Table 1: Test case setup for four different test sets with increasing complexity.

Input/Set	Set 1	Set 2	Set 3	Set 4
Number of Group Properties	5	10	25	50
Number of Balance Functions in Objective Function	5	10	25	50
Number of Balance Functions in Constraint	1	2	4	10
Number of Parameters	10	25	50	100

Based on these values the Group Properties, Balance Functions, Parameters and the constraints were created.

The Group Properties are generated as a vector with random numbers in the interval [-1, 1]. The step that the adaptation model normally would do to create these properties from the role distribution and the properties of the roles is omitted since this is done only once at the beginning and therefore does not influence the complexity of the problem. Furthermore, different value ranges for different properties are not considered as these are not significant in the developed model due to the normalization of the properties.

For the parameters three sets are required: the upper and lower bounds and the default values that can be used as starting point for the algorithms. First the lower bound is generated as a vector with random values in the interval [-1, 1]. Then the default values for the parameters are created by adding a random vector with values in the interval [0, 1] on the lower bounds. The upper bounds for the parameters are created in the same way, by adding a vector on the default values. The bounds on parameters are added as linear inequality constraints to the problem.

The Balance Functions are created by randomly picking up to 20 percent of the available parameters. These are then one by one concatenated using multiplication and summation. Each of these have a 45 percent chance of being used. With a ten percent chance the chosen parameter is multiplied with a random Group Property before being concatenated using summation. This accounts for nonlinear relationships between tasks and Group Properties. These probabilities have been chosen, as an even distribution of additive and multiplicative effects is expected to be representative. Nonlinear relationships only have a very low probability as they are expected to be relatively rare.
In this state the relating functions are also used as nonlinear constraints to model the task bounds. The task bounds are vectors of randomly generated number in the interval [-100, -1] for the lower and [1, 100] for the upper bounds. The values for the bounds are given this large value range to keep as many generated problems as possible solvable.

For use in the target function, the Balance Functions are expanded by subtracting one Group Property from each of the functions to model the linear relationship between tasks and group properties.

5.2 Algorithm Set-Up

Three different solvers available in MATLAB have been chosen for the evaluation.
Fmincon employs deterministic algorithms to solve constrained nonlinear optimization problems. It is a gradient based method, which requires the objective function and the constraints to be continuous and have continuous first derivatives. Since Fmincon works with gradients, it will most often stop in a local minimum, without any means of determining if it is also a global minimum. This can be problematic, when there are many local minima that are significantly worse than the global minimum. Also, this solver cannot be used if the adaptation model incorporates balance functions that are not

continuous or whose derivatives are not continuous, or if parameters need to be constrained to integer values. As mentioned in previously however, most games should not require these additional constraints. This solver offers different algorithms to choose from. All of these that were applicable to our problem were evaluated.

The second solver chosen is the Genetic Algorithm. While genetic algorithms tend to take more time, they have a better chance of finding a global minimum than Fmincon when several local minima are present in the problem. Also, in case that some mechanics cannot be modelled with continuous functions, the genetic algorithm is still applicable, as it does not rely on gradients. It can also deal with integer valued parameters. So it might be a good choice when these more complex problems need to be addressed. In this evaluation they are however not considered, as they are seen as special cases. The genetic algorithm implemented in the Global Optimization Toolbox offers several settings that influence how well the algorithm performs with different problems, such as fitness scaling and ranking functions, selection function and hybrid functions that try to refine a solution found by the genetic algorithm.

To evaluate if any of the options have significant influence on the performance of the genetic algorithm, the algorithm was run with each test set once for each possible option choice with its default choice for the other options. With regard to the efficiency requirement, a maximum time limit of 30 seconds is set for the genetic algorithm to ensure that it finishes in reasonable time for application during level loading. Additionally the population size is set to 50 to ensure that the algorithm does not take too much time for each iteration and can thus perform several iterations within the time limit.

The third solver that has been chosen for the evaluation is Simulated Annealing. This is another heuristic solver. This algorithm has been chosen as it works differently from the genetic algorithm and might therefore perform differently on the developed model. The MATLAB implementation of simulated Annealing does not directly allow constraints. This is compensated in this evaluation by adding the constraints as a penalty term to the objective function. Equality constraints can be transformed into a penalty term by calculating their absolute value. Inequality constraints can be transformed by calculating the maximum over the constraints and zero. As a penalty factor 100 has been chosen. For simulated annealing three options are considered: the annealing function that specifies how new points are generated, the temperature update function that changes the temperature after each step and an optional hybrid function to improve the result found by simulated annealing.

Since the amount of options is much smaller than for the genetic algorithm, a pairwise approach was chosen to ensure that every option has at least once been combined with every other option. This resulted in nine test cases. Additionally, to adhere to the efficiency requirement, a time limit of thirty seconds and a maximum of 1000 iterations as well as a maximum number of function evaluations of 3000 are set.

5.3 Evaluation results

For Fmincon, the algorithms Interior Point and SQP behaved similarly concerning execution time and quality of solution. Active Set however found better solutions as shown in Figure 2 but also required more time to do so.

For the Genetic Algorithm, using a hybrid function to refine the solution expanded the execution time, while not improving the overall solution quality. Significant differences for the quality of the solutions can only be found for the largest test cases. Employing Top Scaling delivered the best quality for these test cases. The average performance values are shown in Figure 3.

For Simulated Annealing, the greatest influence on execution time and quality is the used hybrid function. Employing Fmincon to improve the solution slightly increases execution time and greatly increases the quality, while Patternsearch greatly increases the execution time while only slightly improving the quality of the solution. The respective performance values are shown in Figure 4.

Comparing the different solvers concerning the quality of the solution as depicted in Figure 5 it can be seen that Simulated Annealing and some of the variants of the Genetic Algorithm provide the best results. For Fmincon only Active Set manages to find a solution comparable to the worse variants of the Genetic Algorithm, while the other variants find strictly worse solutions. The reason for this will

most likely be the presence of global minima in the problems. While Fmincon is good at finding a local minimum fast, it has no means to distinguish a local minimum from a global one. The genetic algorithm and simulated annealing have a better chance of finding a global minimum or a better local minimum due to their random exploration.

The results of the evaluation show that there is a conflict between execution time and quality of the solution. While Fmincon, when using SQP or Interior Point, clearly outperforms the other solvers when it comes to execution times, it provides solutions of lower quality than the other two solvers. Simulated annealing and the genetic algorithm provide better solutions, but at the cost of execution time. While simulated annealing provides as good a solution as the genetic algorithm, one has to keep in mind, that simulated annealing in this set-up does not ensure that all constraints are fulfilled. The decision which solver is best to use comes down to a case by case basis, based on whether low computation time or high quality results are more important for the game. For example, a game that requires online adaptation, e.g. because the players can customize their avatars on the fly, the use of Fmincon would certainly satisfy the time requirements and the lower quality of the individual adaptations might be not as detrimental when the solution is constantly updated by running the optimization again. On the other hand, if a game is only adapted once during loading, the higher computation time of the genetic algorithm will be less of a problem, but the higher quality of the solution will be important. Figures 5 and 6 show the solution quality and the execution times for one variant of each solver in comparison. The variants are SQP for Fmincon, Genetic Algorithm with Top Scaling and Simulated Annealing employing Fast Annealing and Fmincon as a hybrid function.

Another aspect to consider when choosing which algorithm to use is the transition model proposed in the previous section. This evaluation showed that the size of the problem greatly influences the computational time required to find a solution. Also for the smaller sets, the difference in quality of the solution found was much smaller and the overall quality was higher. This difference in quality might be based on the smaller problem space that needs to be searched for a solution as well as a reduced number of minima because of the reduced problem space. In any case, splitting up a level will greatly benefit the computational time required by the adaptation. As can be seen from the graphs, the execution time does not rise linearly with the size of the problem but steeper. This indicates that if for example a level can be split up into two segments that define problems with only half the size as the entire level, solving both segments would be achieved faster than solving the larger problem for the entire level. This should especially be considered in an online adaptation scenario. The effectiveness of using the transition model will ultimately depend on how much the size of the problems of the parts of a level can be reduced in size compared to the problem of the entire level.

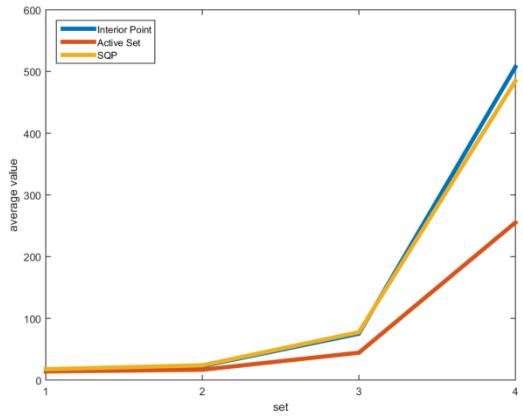

Figure 2: Average solution values for all Fmincon algorithms.

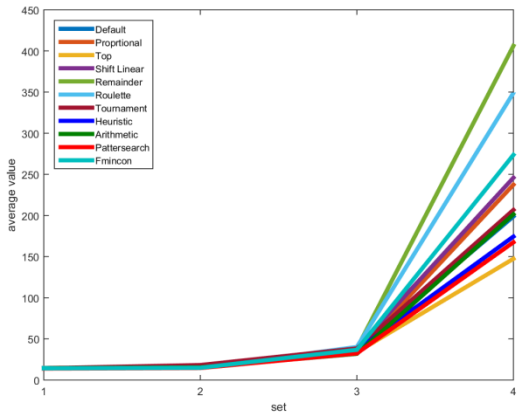

Figure 3: Average solution values for all genetic algorithms.

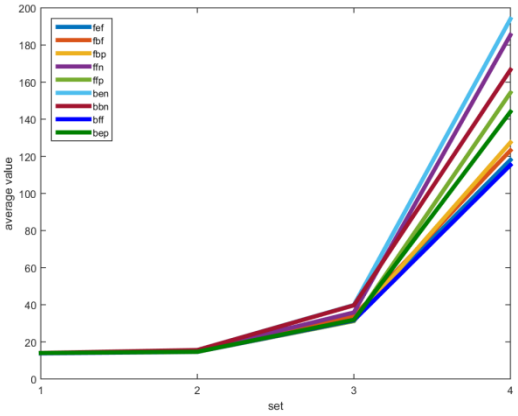

Figure 4: Average solution values for all Simulated annealing algorithms.

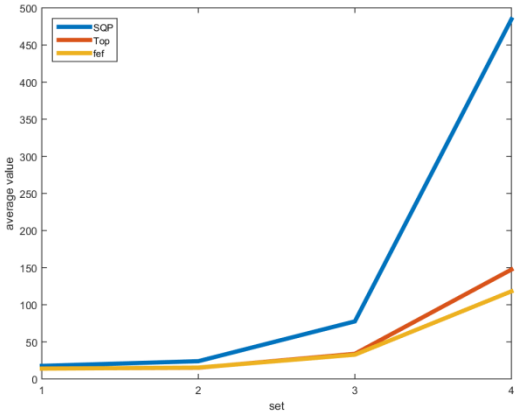

Figure 5: Average solution values for best performing algorithms for each solver.

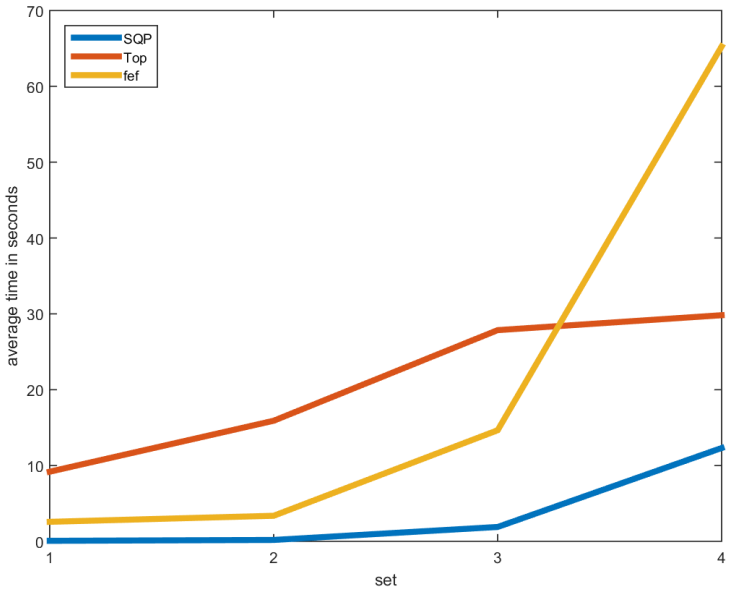

Figure 6 Average computation time for the algorithms, performing best in average solution value.

6. Conclusion

In this paper an adaptation model for cooperative role-based games was developed. This is done by using mathematical formalization of roles, groups and tasks pertaining to these roles and groups. Based on this mathematical foundation the adaptation model was developed as an optimization problem.

After establishing the properties and restrictions of the optimization problem it was implemented in MATLAB and three different solvers, Fmincon, Genetic Algorithm and simulated Annealing, provided by the (Global) Optimization Toolbox were evaluated for their performance on the model. All solvers proved to have their advantages for certain situations. Fmincon is most useful when real-time adaptation is required. If a little more time is available and bounds placed on tasks and parameters of the problem are not strict, Simulated Annealing can be used to find better solutions. The Genetic Algorithm can be used if the time requirements are not as important but a high quality solution and satisfaction of bounds is important.

In future work the model will be applied to multiple use cases with group properties and tasks found for many current MMORPGs. Expanding on the generic properties and tasks complex boss fights with many existing roles can be modelled and adapted for all available group compositions.

References

J. W. Chinneck. (2006) Practical Optimization: a Gentle Introduction. *Systems and Computer Engineering*, Carleton University, Ottawa.

Scott Kirkpatrick, C. Daniel Gelatt, Mario P Vecchi, et al. (1983) Optimization by simulated annealing. Science.

J. Konert, D. Burlak, S. Göbel, R. Steinmetz. (2013) GroupAL: ein Algorithmus zur Formation und Qualitätsbewertung von Lerngruppen in E-Learning-Szenarien mittels n-dimensionaler Gütekriterien. *DeLFI Vol 13*.

O. Missura and T. Gärtner. (2009) Player Modeling for Intelligent Difficulty Adjustment. *Discovery Science*, Springer, pp 197-211.

C. Pedersen, J. Togelius and G. N. Yannakakis. (2009) Modeling Player Experience in Super Mario Bros. *IEEE Symposium on Computational Intelligence and Games*. CIG 2009. pp 132–139.

C. Pedersen, J. Togelius and G. N. Yannakakis. (2010) Modeling Player Experience for Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(1), pp 54–67.

J. Togelius, E. Kastbjerg, D. Schedl, and G. N. Yannakakis. (2011) What is Procedural Content Generation? Mario on the borderline. *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. ACM.

T. Tregel, C. Reuter, S. Göbel, R. Steinmetz. (2016) Generating Multiplayer Games for Interaction Learning using Game Design Patterns. *European Conference on Games Based Learning*. Academic Conferences International Limited.

D. Whitley. (1994) A Genetic Algorithm Tutorial. Statistics and Computing, 4(2).

G. N. Yannakakis and J. Togelius. (2011) Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, 2(3), pp 147–161.