Neuroevolutionary Constrained Optimization for Content Creation

Antonios Liapis, Georgios N. Yannakakis, *Member, IEEE* and Julian Togelius, *Member, IEEE*

***Abstract*— This paper presents a constraint-based procedural content generation (PCG) framework used for the creation of novel and high-performing content. Specifically, we examine the efficiency of the framework for the creation of spaceship design (hull shape and spaceship attributes such as weapon and thruster types and topologies) independently of game physics and steering strategies. According to the proposed framework, the designer picks a set of requirements for the spaceship that a constrained optimizer attempts to satisfy. The constraint satisfaction approach followed is based on neuroevolution; Compositional Pattern-Producing Networks (CPPNs) which represent the spaceship's design are trained via a constraint-based evolutionary algorithm. Results obtained in a number of evolutionary runs using a set of constraints and objectives show that the generated spaceships perform well in movement, combat and survival tasks and are also visually appealing.**

I. Introduction

Design and production of game content is amongst the most costly and labor intensive parts of the game development process. The creation of content supported by algorithmic means may help to overcome the content creation bottleneck while also allowing for novel content to be generated and tested faster, game replayability to be increased and design efforts to be pushed to the limits of human creativity. However, the utility of content for a particular game and a specific player is far from obvious if content is not linked to computational models of player experience and if efficient search algorithms are not generating the required novelty.

Inspired by the experience-driven procedural content generation (EDPCG) framework of Yannakakis and Togelius [1] we propose a search-based constrained optimization PCG [2] approach to spaceship design, in particular hull shape design and weapon/thruster topology design. The paper introduces a generic framework which optimizes a spaceship for movement, combat and survival competencies given a set of design requirements, a physics simulator of a particular game engine and a set of steering behaviors. The PCG approach proposed evolves Compositional Pattern-Producing Networks (CPPNs) [3] via the Feasible-Infeasible Two-Population (FI-2Pop) [4] genetic algorithm which yields a neuroevolutionary constraint satisfaction algorithm.

Results obtained through the study presented in this paper suggest that the proposed framework is able to design spaceships that achieve high performance in terms of combat, movement and survival. While ad-hoc designed solutions are performing well, evolution of spaceships allows for adaptivity and offers more flexibility and creativity in content design. The evolved spaceships are not only able to perform well in certain tasks but they are also aesthetically pleasing with unexpected and intriguing shapes.

Authors are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark. Emails: {anli, yannakakis, juto}@itu.dk

The paper is novel in various aspects of computational intelligence and games (CIG) research: it is the first time that a PCG framework is proposed for the creation of content for highly-performing game agents which is independent of physics and motion strategy; it is the also first time PCG is proposed for spaceship design and the first time that CPPNs are combined with FI-2Pop to solve constraint satisfaction problems (for PCG and beyond). While the presented framework is inspired by the Galactic Arms Race game [10], it is distinct in that it evolves the spaceships themselves rather than their weapons, controls the generative process through constraints and substitutes the evaluation of content by players (i.e. interactive fitness) with an evaluation of performance during in-game simulations (i.e. simulation-based fitness).

II. Related Work

During the short history of digital game development, procedural content creation has mainly been used for the design of maps, terrains and levels (e.g. in games such as *Rogue*, *Elite* and *Civilization*), though there are examples of item generation such as the recent *Borderlands*. In recent years PCG has gained increased attention from the research community, focusing more on new ways of generating, evaluating and optimizing content.

Search-based Procedural Content Generation (SBPCG), is a special case of the generate-and-test approach to PCG [2] in which the generated content is not simply accepted or rejected but assigned a fitness value guiding the generative process towards new content with higher fitness. Genetic algorithms are ideal candidates for most search-based optimization problems and have been the tool of choice for the majority of SBPCG studies, which range from the generation of game levels [5], [6], [7] and maps [8], [9] to weapon projectiles [10] and game rules [11], [12].

The evaluation of content through simulations (simulation-based PCG) has recently seen several implementations. Togelius et al. [6] estimate the fitness of a track based on the performance of a steering controller modeled after a particular player's driving style. Browne [11] estimates the fitness of evolved board layouts and rules based on numerous aesthetic measurements gathered during a number of self-play simulations. Togelius and Schmidhuber [12] estimate the fitness of a set of game rules based on the game's learnability.

While constraint-based PCG has been implemented in [13], the proposed method is innovative in that a neuroevolutionary approach is employed for constrained optimization of

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

content. Additionally, while CPPNs have been implemented for evolving agent-based content for the Galactic Arms Race game [10], this paper diverges from the approach taken in that game by ensuring (via hard constraints) that the content is functional but also of sufficient quality before making it available to the player. Unlike the Galactic Arms Race game, content quality in this framework is not assessed by the players, but by metrics gathered during actual in-game simulations testing for a plethora of the spaceships' competences.

III. Neuroevolutionary Constrained Optimization

This section presents the two main components of the neuroevolutionary constrained optimization algorithm used for the purposes of this study.

A. CPPN-NEAT

Introduced by Stanley [3], Compositional Pattern-Producing Networks (CPPNs) are neural networks specifically designed to represent content with regularities, and which are capable of being optimized through artificial evolution. Assuming that development in nature consists of a series of progressively more localized *coordinate frames* (where a coordinate is a "conceptual device for describing an abstract configuration of any type" [3]), Stanley argues that development is analogous to a series of function compositions which transform the base coordinate frame to increasingly more localized coordinate frames with each transformation applied. This sequence of function compositions can be represented as a connected graph of such functions, with the initial coordinate frame as input and the most localized coordinate frames as output.

CPPNs can be optimized via neuroevolution of augmenting topologies (NEAT) [14]. NEAT starts evolution with a uniform population of CPPNs with the simplest topology (no hidden nodes) and random connection weights. As evolution progresses, more hidden nodes and links are added to the CPPNs; when a node is first added to the network, its activation function is selected randomly from a range of pattern producing functions (such as symmetrical or periodic functions). Genetic diversity is maintained through *speciation*, with individuals competing primarily with members of their own species, allowing them to optimize their structure without being overwhelmed by individuals of different species with more complex (and possibly more optimal) topologies.

B. FI-2Pop

While genetic algorithms have shown great promise in the domain of function optimization [15], the difficulties they face in solving constrained numerical optimization problems [16] has given rise to many different methods for handling such problems. The Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm [4] is a recent approach to constrained optimization through artificial evolution, its principle being the maintenance (throughout the execution of the algorithm) of two populations — one containing only feasible individuals and the other containing only infeasible. Each population selects and breeds only among its own members in order to optimize its fitness function, with each population having a different evaluation strategy.

While the feasible population conducts its optimization in much the same way as in an unconstrained problem, the objective function of the infeasible population shifts the latter towards the boundary of feasible space, where the optimum solution often lies [16]. The proximity of infeasible individuals to the boundary of feasible space increases their chances of producing feasible offspring. The offspring of both generations are tested for constraint satisfaction, with infeasible offspring (regardless of whether their parents were feasible or infeasible) being inserted into the infeasible population and feasible offspring being inserted into the feasible population. This migration of offspring from one population to the other (an indirect form of inter-breeding) contributes to the variation of both populations; depending on the size of the feasible set, this migration may be the only source for feasible individuals.

The algorithm proposed in this paper evolves CPPNs via FI-2Pop yielding a constrained optimization approach through neuroevolution.

IV. Methodology

The type of game content evaluated and optimized is a single two-dimensional polygon representing the hull of a spaceship, including weapons or thrusters attached to the edges of the spaceship's hull. The spaceship must fulfill some minimum requirements, and is evaluated based on its performance in a series of simulations of typical shooter game circumstances. This section presents the process of the spaceship's generation and its evaluation.

A. Representation

The patterns in the spaceships' hulls are represented via a CPPN (see Fig. 1). The CPPN receives a sequence of inputs in the form of 2D coordinates (corresponding to 64 equidistant points on a circle) and returns a sequence of outputs corresponding to the points of the spaceship hull's pattern. Each output vector consist of the 2D coordinates X and Y of each point and a γ value which indicates if the point has a weapon or thruster attached. This sequence of outputs is translated into the spaceship's hull coordinates as well as specific weapon and thruster types according to a collection of game-specific parameters. These game parameters include a collection of weapon types and thruster types and constants such as mass per surface unit and maximum width and height of a spaceship (see (c) in Fig. 1). An output vector's X and Y values are used directly as the coordinates of the corresponding point on the spaceship's hull while the continuous γ value is translated into different categorical weapon types and thruster types (each with their own parameters such as cost, thruster power, weapon cooldown or projectile damage). If $\gamma < -0.7$ then the point has a thruster attachment, the type of which depends on the value range each thruster is

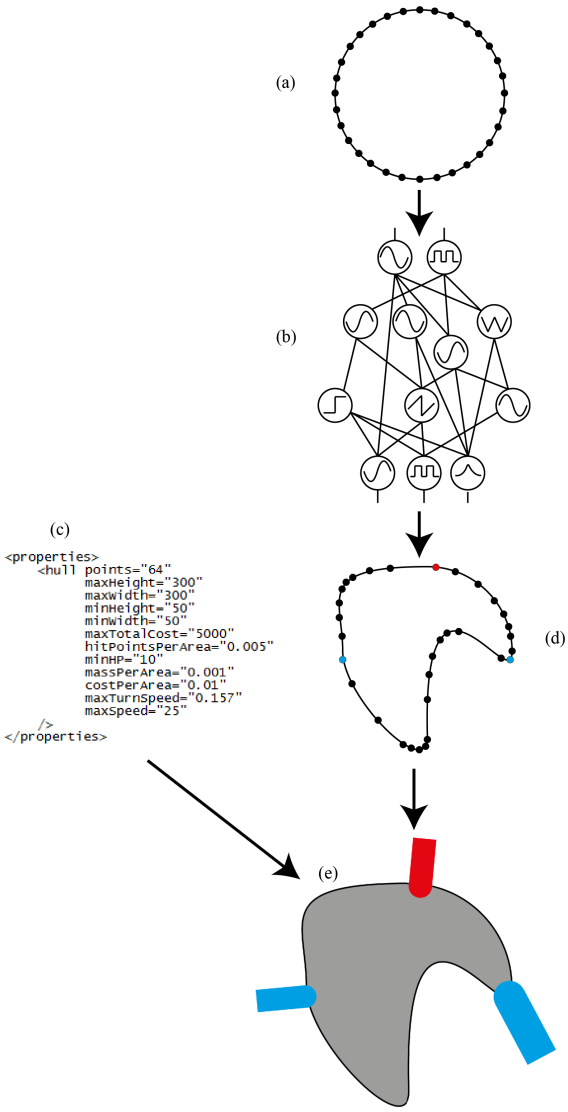


Fig. 1. Generation of a spaceship (e) from the input circle (a), the CPPN (b) and the game parameters (c). The red element represents a weapon (type *Laser*) and the two blue elements represent thrusters of different types.

assigned within $[-1, -0.7)$. If $\gamma > 0.7$ then the point has a weapon attachment, the type of which depends on the value range each thruster is assigned within $(0.7, 1]$. For all other γ values the point has no attachment.

Once the spaceship has been generated from the sequence of outputs of the CPPN, its physical properties are determined based on its surface area, its centroid and the alignment of thrusters with regards to it. The physical properties determined this way include the spaceship's mass, its thrusters' force and torque (caused by thrusters misaligned with the centroid). The spaceships' physics model is elaborate and allows for speed and acceleration to have different directions and enabling momentum.

B. Constraint satisfaction

Spaceships must fulfill some minimum requirements which are linked to the graphics engine, the physics simulation or the game environment. Constraints can also be

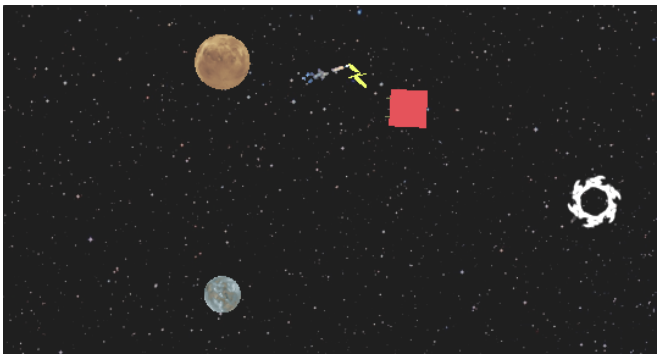


Fig. 2. A screenshot of a rendered simulation, showing all the available components. Depicted are the procedurally-generated tested spaceship (yellow), an enemy spaceship (red) loaded from a library of predetermined designs, two planets acting as obstacles (the two spheres) and the goal area (white) towards which the test spaceship is moving.

imposed on performance by the designer, such as a maximum time required to complete a task. A set of hard constraints is checked before the spaceship is evaluated via game test simulations for performance; game test simulations are skipped if those constraints are violated. Such hard constraints arise from the needs of rendering physics simulations (such as a non-degenerate polygon and positive mass), from the need of a spaceship appearing "plausible" to the user (such as weapons and thrusters that do not intersect with each other) and from the game design itself (which can impose a maximum number of weapons and thrusters, or a maximum speed). In addition, some game test simulations constitute additional hard constraints (e.g. maximum time to complete or maximum deaths).

Given the game- or author-imposed requirements the problem of spaceship design is clearly one of constrained optimization. The Feasible-Infeasible Two-Population genetic algorithm (presented in section III-B) is capable of handling such a problem by maintaining both a feasible and an infeasible population. The feasible population consists of individuals corresponding to spaceships which fulfill all the constraints, and strives to maximize its members' fitness in a set of game simulations (detailed in section IV-C). On the other hand, the infeasible population consists of individuals corresponding to spaceships which violate one or more constraints, and strives to minimize its members' distance from feasibility which is calculated as a sum of each constraint's distance from the feasible solution. While some constraints have a scalar distance from feasibility (e.g. number of degenerate segments) other constraints have a boolean distance from feasibility (e.g. simulation-dependent constraints can be either passed or failed). The fitness score of the infeasible population is inversely proportional to the total distance from feasibility, providing sufficient bias towards individuals which only fail a small number of constraints.

C. Evaluation of performance

Spaceships must be able to perform the tasks as those are imposed by the game designer, ideally in the most

efficient fashion. The performance of spaceships is evaluated based on their behavior in a series of simulations of typical game situations. A simulation takes place in an infinite 2D world consisting of goals, planets (acting as obstacles) and spaceships (see Fig. 2). Apart from the tested spaceship, each simulation identifies the number and properties of allied and enemy spaceships as well as the steering strategy applied to their controllers. The steering controller may combine numerous steering behaviors such as seek, arrive, target, target-approach, avoid, and flee. Inspired by Reynolds' steering behaviors [17], the steering controller accounts for the spaceships' elaborate physics model (with momentum, misaligned thrusters and the aiming conditions of diverse weapon types). The steering behaviors are designed so that most generated spaceships can be controlled, but may be more suited to specific spaceship setups; performance optimization aims to discover spaceship designs which benefit the most from the given steering controller and physics model.

The simulations test the ability of a spaceship to reach a goal (with or without obstacles along its path) or destroy enemy spaceships. When the winning condition of a simulation is satisfied, the performance of the spaceship is assessed based on in-game statistics (metrics) gathered during the simulation. Each test yields a test score T and assesses one (or more) of the three following competencies: *movement*, *combat* and *survival*. The test scores of the various simulation tests are aggregated into a sum normalized to $[0, 1]$ which represents the fitness value of feasible individuals. The fitness function, f, is calculated as:

$$f = \frac{\sum_{i=1}^{N} T_i}{\sum_{i=1}^{N} M_i}$$

where N is the number of different simulation tests (7 in this paper); T_i is the test score for the i-th test and M_i a constant equal to the maximum possible test score for the i-th test. The proposed normalization gives those performance tests with more gathered metrics a higher impact on the fitness score than simpler tests, which have fewer metrics.

The simulations used to assess performance in this paper are presented below along with a type for their test scores. Since many simulations share the same performance metrics, any variable name in the following types has the same definition as in the first type where it was presented.

1) **Movement:** A basic speed test consisting of the test spaceship and a goal area, which measures the efficiency with which the spaceship reaches the goal. Its test score is:
$$T_1 = \frac{d}{t \cdot S_{max}} + \frac{d_{start}}{d}$$
where d is the trail distance (the total distance between consecutive positions of the test spaceship); S_{max} is the maximum allowed speed; t is the simulation's duration; and d_{start} is the shortest distance between the spaceship's starting position and the goal. $M_1 = 2$.
2) **Movement:** A basic collision test consisting of the test spaceship, a goal area and a planet between the two, which measures the efficiency with which the spaceship reaches the goal while avoiding collisions. Its test score is:
$$T_2 = T_1 + \frac{t_c}{t}$$
where t_c is the total duration that a spaceship was colliding with an obstacle. $M_2 = 3$.
3) **Movement:** A complex collision test including the test spaceship, a goal area and five planets scattered around the path between the spaceship and the goal. This test measures the efficiency with which the spaceship reaches the goal while avoiding collisions in a more complex environment. Its test score is:
$$T_3 = T_2 + \frac{d_{start}}{d_{end}}$$
where d_{end} is the distance between the spaceship's final position and the goal. $M_3 = 4$.
4) **Combat:** A basic shooting test consisting of the test spaceship and an immobile enemy spaceship without thrusters or weapons. This test measures the efficiency with which the spaceship destroys the enemy, and its test score is:
$$T_4 = \frac{D_e}{t_{co} \cdot D_{max} \cdot W_{max}} + \frac{D_e}{D_s}$$
where D_e is the amount of damage inflicted to enemy spaceships; t_{co} is the duration of the combat; D_{max} is the highest damage per unit of time among all possible weapons; W_{max} is the maximum number of weapons allowed on a spaceship; and D_s is the total damage of shots fired from the test spaceship. $M_4 = 2$.
5) **Movement and Combat:** A chasing test consisting of the test spaceship and a fleeing enemy spaceship without weapons, which measures the efficiency with which the spaceship can reach the enemy and destroy it. Its test score is:
$$T_5 = T_4 + \frac{d}{t \cdot S_{max}} + \frac{D_e}{H_e}$$
where H_e is the enemy's initial health points. $M_5 = 4$.
6) **Movement and Combat:** A chasing test as (5) but with a faster enemy spaceship. Its test score T_6 is calculated as T_5 and $M_6 = M_5$.
7) **Movement, Combat and Survival:** A shooting test which includes the test spaceship and an enemy spaceship with weapons and thrusters and with the same behavior as the test spaceship (moving towards and shooting its enemies while avoiding collisions with them). This test measures the efficiency with which the spaceship can destroy an aggressive, competitive enemy.
$$T_7 = T_4 + \frac{D_e}{H_e} + \frac{s_r}{s_f}$$
where s_r and s_f are the number of shots received by the test spaceship and fired by the enemy, respectively. $M_7 = 4$.

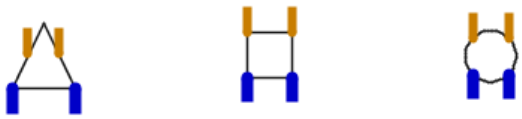


Fig. 3. Ad-hoc baseline solutions for spaceship performance: a triangle, a square and a circle. Blue elements depict thrusters while yellow elements depict weapons. The surface of each spaceship's hull corresponds to a mass that, with the given thrusters, can reach a speed equal to the maximum speed allowed by the game environment.

TABLE I

FITNESS OF THE THREE BASELINE SOLUTIONS

Tests used to calculate fitness	Square	Triangle	Circle
Tests 1 to 3 (M test cluster)	0.9236	0.9294	0.9330
Tests 1 to 6 (M/C test cluster)	0.7333	0.7383	0.7461
Tests 1 to 7 ($M/C/S$ test cluster)	0.7288	0.7320	0.7385

Combining all of the above simulation tests allows for most functionalities expected from a spaceship in a spaceship combat game to be evaluated; the cluster of tests 1 to 7 is labeled *M/C/S test cluster*. By using the tests described under (1), (2) and (3), evolution optimizes only the spaceships' competency at movement (focusing on speed and maneuverability), rendering the presence of weapons unnecessary; the cluster of tests 1 to 3 is the *M test cluster*. Combining all tests except (7), evolution disregards the spaceships' survival, focusing on movement and combat efficiency; the cluster of tests 1 to 6 is the *M/C test cluster*.

V. EXPERIMENTS

This section presents a set of ad-hoc designed baseline solutions to the chosen tests and concludes with experiments of neuroevolutionary constrained optimization.

A. Baseline Solutions

Some obvious ad-hoc baseline solutions to spaceship design include a triangle, a circle and a rectangle as shown in Fig. 3. Those three designs are used for comparative purposes against our approach. The hulls' dimensions are chosen so that the maximum speed (which constitutes a game-specific constraint) can be reached with the given thrusters. Since speed is a major contributor of performance in most of the simulation tests described in section IV-C, it is expected that spaceships on the edge of feasibility with regards to the speed constraint will have very high fitness scores.

Table I shows the fitness value of the three ad-hoc shapes for the three different clusters of simulation tests described in section IV-C. As expected, the fact that all ad-hoc solutions have similar hull areas (and therefore similar mass and maximum speeds) results in similar fitness values regardless of the shape of the spaceship; any fitness difference can be accounted to the difference in the distance of weapons from each other and slight disparities in mass due to rounding.

B. Neuroevolutionary Spaceship Design

The generative process for spaceships through a CPPN, as detailed in Section IV-A, allows for any number of weapons and thrusters to be attached to a spaceship's hull. For the purposes of comparing the baseline solutions with evolved spaceships, the number and types of weapons and thrusters are predefined as part of the CPPN's inputs, and the γ output is disregarded — the weapons and thrusters of generated spaceships are the same (both in number and type) as those used for the baseline solutions of Fig. 3. This *predefined* representation has a more limited search space than the one presented in Section IV-A, but also manages to generate spaceships that always satisfy numerous constraints detailed in Section IV-B. This results in a large feasible set with regards to the search space, increasing the likelihood of feasible offspring from both feasible and infeasible parents.

1) Predefined representation: After 200 generations of the CPPN-FI-2Pop algorithm using a fitness calculated on all the simulations (*M/C/S* test cluster), the fittest spaceship generated by this predefined representation has a mean fitness of 0.7284 (stdev is 0.0063) in 5 individual runs. The fittest spaceship throughout these runs has a fitness value of 0.7384. When this spaceship is tested only on the M test cluster or the M/C test cluster, its fitness is 0.9121 and 0.7303, respectively.

The optimized spaceship's fitness values are very close to those of the ad-hoc solutions; evolution manages to find individuals on the edge of the feasible space, achieving a speed very close to the maximum allowed. The fact that the number and types of weapons and thrusters are predefined ensures to a large extent that the generated individuals will be feasible (and therefore able to complete all simulations) but also narrows the spaceships' performance to a limited value range (with boundaries on both the minimum and the maximum performance). Section V-B.2 will illustrate that allowing for different weapon and thruster topologies in the same population allows for a larger search space and ultimately higher fitness values than the ones presented here.

Fig. 4 illustrates the progress of the best individual's fitness throughout the optimization process, broken down by the contributing competencies in different types of test levels (as identified in section section IV-C). the figure also shows the phenotypes of the best individual on different stages of evolution (every 40 generations). Since most simulations evaluate Movement or Movement and Combat, they have the highest contributions to the fitness score. The figure shows that the population includes highly fit individuals even in the initial population. Within the first 40 generations, the best individual changes often, mostly increasing the movement composite of performance — around the 40th generation the movement composite has reached its maximum value with the given constraint on maximum speed. After the first 40 generations the best individual changes only in three instances, mostly increasing its competency in the Movement, Combat and Survival simulation. The final best individual does not have a very different fitness value from those in early stages of evolution.

2) Standard representation: Using the standard representation as detailed in section IV-A, the generated spaceships

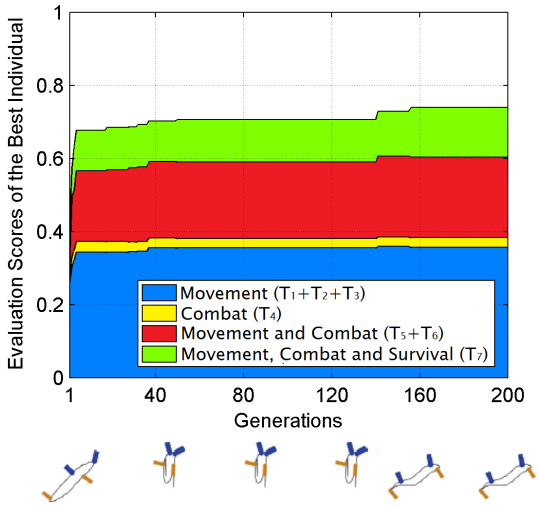


Fig. 4. Stacked area plot of the performance optimization on all simulations (the $M/C/S$ test cluster) using the predefined representation. The session has a total population of 500 individuals. In the phenotypes shown, blue and yellow elements depict thrusters and weapons, respectively.

may have any number of thrusters and weapons. The number of thrusters cannot increase the speed of the spaceship, which is constrained to a maximum, but the number of weapons can increase the combat efficiency of the generated spaceships over that of the ad-hoc shapes of Fig. 3.

Given the freedom of this representation with regards to attached weapons and thrusters, the search space is very large, while the numerous constraints on the maximum number of weapons and thrusters, the maximum speed and the requirements of the individual simulations result in a small feasible set. This greatly affects the number of feasible individuals, since neither feasible nor infeasible parents are likely to generate feasible offspring; the resulting small feasible population hinders the optimization of performance. The fitness values of the best evolved individuals, averaged across 5 individual runs, are displayed in Table II. When optimizing only for the movement competency (the M test cluster), the best optimized individual among these runs has a slightly higher fitness than that of the baseline solutions of Table I — the constraint on maximum speed does not permit significantly higher performance. When combat (or combat and survival) is included in the fitness score, the best optimized individual has much higher fitness than the baseline solutions, since the weapon topology plays a significant role in the targeting behavior and the combat efficiency of the spaceship. Since the standard representation can generate spaceships with a larger number of weapons than the baseline solutions, the comparison between them is not entirely fair. However, the challenge a human designer faces at guessing such an optimal weapon topology supports the simulation-based optimization approach.

The progress of the best individuals' fitness throughout the optimization process in each of the three different test clusters is shown in Fig. 5. Where applicable, these figures illustrate the progress of the best individual's contributing competencies in different types of test simulations (as

TABLE II

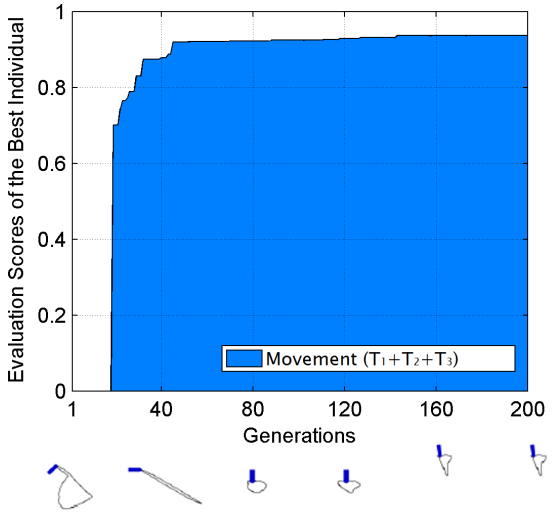
STANDARD REPRESENTATION: FITNESS OF THE BEST EVOLVED INDIVIDUALS AFTER 200 GENERATIONS OUT OF 5 DIFFERENT RUNS

Tests used to calculate fitness	Mean	St. Dev.	Max
Tests 1 to 3 (M test cluster)	0.9321	0.0043	0.9367
Tests 1 to 6 (M/C test cluster)	0.8023	0.0086	0.8147
Tests 1 to 7 ($M/C/S$ test cluster)	0.7117	0.0930	0.7986

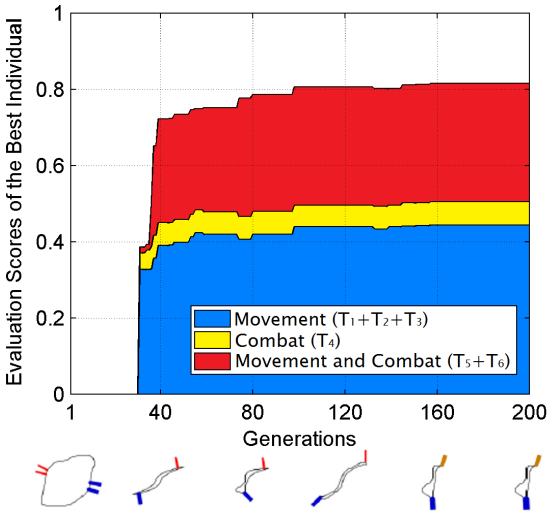
identified in section section IV-C). Under each figure, the phenotypes of the best individual on different stages of evolution are shown; the first phenotype always corresponds to the first feasible individual in the population, while the other phenotypes are chosen on a 40 generation interval.

When optimizing only the movement competency (the M test cluster), it should come as no surprise that the phenotypes do not have any weapons, since those are unnecessary for completing the provided simulations. Fig. 5(a) illustrates how the large first feasible spaceship is quickly optimized to a leaner, lighter one. Within the first 30 generations after the first feasible spaceship, the best individual has managed to achieve a high fitness score; in the remaining generations only small changes occur in the phenotypes favoring small and compact spaceship hulls to minimize torque and the likelihood of collisions. With the inclusion of combat (see Fig. 5(b)) the discovery of the first feasible individual in the population becomes more onerous, since it requires the presence of both weapons and thrusters in the corresponding spaceship. The first feasible spaceships are large and slow, and are quickly optimized to increase the movement composite of the fitness by reducing their hull's surface. Not surprisingly, the best individual optimized on the grounds of movement, combat and survival (see Fig. 5(c)) changes often, as it juggles between many different (even conflicting) types of test simulations; the balance of contributions from different tests in the best individual's fitness score changes from one generation to the next. The final best spaceship design is rather unexpected and peculiar. The evolutionary algorithm manages to generate a bizarre hull geometry which permits a thruster to be placed at the front area of the spaceship while a weapon is placed at the bottom area. Given that this peculiar spaceship performs better than the baseline solutions while possessing half the number of weapons and half the number of thrusters (the type of weapons and thrusters are common between the baseline and the optimized solution) indicates that counter-intuitive designs can take better advantage of the steering controller and the physics system than predicted optimal designs.

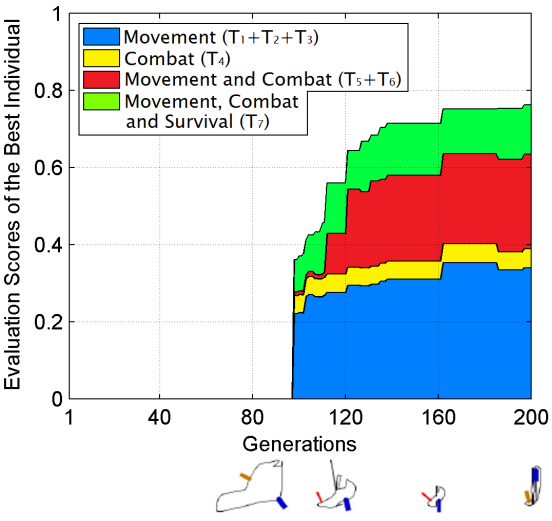
A quick look at the phenotypes of Fig. 4 and Fig. 5 illustrates that search-based optimization can create spaceships of similar or higher fitness than the fitness of hand-crafted solutions. The effort required to reach such fitness values by evolving spaceships seems unnecessary, considering how easy an obvious ad-hoc design solution can be determined; however, these hand-crafted solutions assume prior knowledge of both the test levels and the constraints imposed. Many constraints and straightforward simulations (such as



(a) Optimization on tests 1 to 3 (M test cluster)



(b) Optimization on tests 1 to 6 (M/C test cluster)



(c) Optimization on tests 1 to 7 ($M/C/S$ test cluster)

Fig. 5. Stacked area plots of the performance optimization process, using the standard representation. Each session has a total population of 500 individuals. In the phenotypes shown, blue elements represent thrusters, while yellow and red elements represent different weapon types.

those involving only movement) may have optimal solutions which are immediately obvious to a human designer. However, given the complex steering controller and the physics model used for the spaceships' locomotion, the spaceships' behavior in many simulations is far from obvious.

Neuroevolutionary constraint satisfaction via CPPN-FI-2Pop treats all constraints and all simulations equally: it may require disproportional computational effort to identify optimal spaceships in cases where one is immediately obvious, but can also find optimal spaceships in cases where one is not obvious. Additionally, evolution manages to find spaceships whose hull has more "interesting" shapes than a triangle, a circle or a square. While an "interesting" hull shape does not affect the spaceship's performance, it opens up the potential of evolving spaceships both on the grounds of their performance in simulations and on the grounds of visual appeal.

VI. Discussion

This paper provides a constraint satisfaction PCG framework used to optimize the shape and weapon/thruster topology of spaceships based on actual game simulations. While obvious solutions may be determined by a human designer for clear-cut tasks, the framework is generic across any game physics model and steering strategy allowing it to find optimal or near-optimal spaceships even in cases where such are not obvious a priori. Moreover, the evolutionary process opens the possibilities for optimizing the shape of spaceships not only for performance, but also for visual appeal. By quantifying aesthetic qualities such as symmetry, weight distribution and alignment, the optimization process can create content of high performance incorporating specific visual patterns. That would allow the adaptation of content to a player's individual aesthetic preferences and visual taste while ensuring that it remains both viable (able to perform its required tasks) and of high competitiveness. The process of identifying and quantifying important aesthetic qualities of a spaceship and their optimization process — as well as the specifics for adapting such qualities to a player — is considered for future work.

It should be noted that, while in the current context the framework proposed incorporates a tool for both evaluating and optimizing content, it can also be used solely as a tool for evaluating performance: the spaceships evaluated can be evolved (as those appearing in Table II) or handcrafted (as those appearing in Table I). The benefit of such a tool is that it gives each tested spaceship a score indicative of the required competencies as stated (implicitly) by a human designer. Adjusting an existing simulation or adding a new one is as easy as editing a short XML file, and requires minimal knowledge of the inner workings of the evaluation tool; this allows even game developers with minimal programming skills to assess their created content's performance on specific tasks, which can provide insight on the balance among different spaceships or rank spaceships with regards to challenge (if the spaceships are used as enemies).

The evaluation tool of the framework takes into account both the physics simulations and the steering controller used in the actual game environment, providing a generic approach to performance assessment but also allowing for the re-evaluation of content under new circumstances. In particular, within a game company, hand-crafted or procedurally generated spaceships can be evaluated on an early steering controller; when a more elaborate steering controller or physics simulation is in place the same content can be re-evaluated, providing insight on how the new game environment affects the balance of previously optimized content. Even if the optimization process proposed is not yet entirely suited for the game industry (e.g. for its considerable computational burden and lack of control over the generated results), the evaluation process can still be an adequate authoring tool for game development.

The framework and algorithm proposed are applicable beyond the scope of spaceship design since the potential of constrained optimization opens new possibilities to other applications of PCG. For instance, procedurally generated game levels (such as those for the StarCraft game [8]) can be constrained based on playability or fairness among players, ensuring that feasible maps are both playable and enjoyable, while infeasible maps are iteratively corrected until they become playable. In PCG attempts via interactive evolution (such as Galactic Arms Race [10]), the inclusion of constraints on the performance competencies of the generated content can ensure that the content presented to the player is at least functional. The current argument for interactive evolutionary content is that players will identify and discard dysfunctional content [18]; however, making dysfunctional content available to the player in the first place is arguably unnecessary, leading only to player frustration and fatigue while enhancing the impression that content is generated on a hit-or-miss basis. Constrained optimization can be combined with interactive evolution, with the infeasible population being optimized based on a heuristic measuring the distance from feasibility and the feasible population being optimized based on a player's implicitly inferred or explicitly reported preferences [1].

VII. Conclusion

This paper presented a novel method where search-based procedural content generation is performed utilizing constrained optimization techniques. The satisfaction of a priori defined and simulation-dependent constraints ensures that feasible content not only satisfies technical requirements but also game design choices, and is able to fulfill all the required tasks (if not in the most efficient manner). The efficiency at fulfilling the tasks required from the content is assessed through simulations using a physics system and steering behaviors specific to a game environment. Combining FI-2Pop with CPPN-NEAT, both the feasible individuals are optimized towards better performance and the infeasible individuals are brought closer to the boundary with the feasible space. Findings demonstrate that spaceships generated through the CPPN-FI-2Pop framework are not only more visually "interesting" than those designed by humans, but manage to be even more competitive than their hand-crafted counterparts.

References

[1] G. N. Yannakakis and J. Togelius, "Experience-driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, 2011, (in print).

[2] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," in *Proceedings of the EvoStar Conference*. Springer-Verlag, April 2010.

[3] K. O. Stanley, "Exploiting regularity without development," in *Proceedings of the AAAI Fall Symposium on Developmental Systems*. Menlo Park, CA: AAAI Press, 2006.

[4] S. O. Kimbrough, G. J. Koehler, M. Lu, and D. H. Wood, "On a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch," *European Journal of Operational Research*, vol. 190, no. 2, pp. 310–327, October 2008.

[5] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling Player Experience in Super Mario Bros," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Milan, Italy: IEEE, September 2009, pp. 132–139.

[6] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, 2007, pp. 252–259.

[7] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelback, and G. Yannakakis, "Multiobjective exploration of the starcraft map space," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, 2010, pp. 265 –272.

[8] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis, "Multiobjective exploration of the starcraft map space," in *Proceedings of the IEEE Conference on Computational Intelligence and Games*, Copenhagen, Denmark, 18–21 August 2010, pp. 265–272.

[9] L. Cardamone, G. N. Yannakakis, J. Togelius, and P. L. Lanzi, "Evolving Interesting Maps for a First Person Shooter," in *Proceedings of EvoGames: Applications of Evolutionary Computation*, ser. Lecture Notes on Computer Science, vol. 6624. Springer, 2011.

[10] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Evolving content in the galactic arms race video game," in *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 241–248.

[11] C. Browne and F. Maire, "Evolutionary game design," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 1, pp. 1–16, mar. 2010.

[12] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, dec. 2008, pp. 111–118.

[13] N. Sorenson and P. Pasquier, "Towards a generic framework for automated video game level creation," in *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, vol. 6024. Springer LNCS, 2010, pp. 130–139.

[14] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[15] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," in *Proceedings of the 4th Annual Conference on Evolutionary Programming*. MIT Press, 1995, pp. 135–155.

[16] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility," in *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1996, pp. 245–254.

[17] C. Reynolds, "Steering behaviors for autonomous characters," in *Game Developers Conference 1999*, 1999.

[18] E. J. Hastings and K. O. Stanley, "Interactive genetic engineering of evolved video game content," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ser. PCGames '10. New York, NY, USA: ACM, 2010, pp. 8:1–8:4.