Constrained Novelty Search: A Study on Game Content Generation

Antonios Liapis anli@itu.dk
Center for Computer Games Research, IT University of Copenhagen, Copenhagen, 2300, Denmark

Georgios N. Yannakakis georgios.yannakakis@um.edu.mt
Institute of Digital Games, University of Malta, Msida, 2080, Malta

Julian Togelius julian@togelius.com
Center for Computer Games Research, IT University of Copenhagen, Copenhagen, 2300, Denmark

Abstract

Novelty search is a recent algorithm geared towards exploring search spaces without regard to objectives. When the presence of constraints divides a search space into feasible space and infeasible space, interesting implications arise regarding how novelty search explores such spaces. This paper elaborates on the problem of constrained novelty search and proposes two novelty search algorithms which search within both the feasible and the infeasible space. Inspired by the FI-2pop genetic algorithm, both algorithms maintain and evolve two separate populations, one with feasible and one with infeasible individuals, while each population can use its own selection method. The proposed algorithms are applied to the problem of generating diverse but playable game levels, which is representative of the larger problem of procedural game content generation. Results show that the two-population constrained novelty search methods can create, under certain conditions, larger and more diverse sets of feasible game levels than current methods of novelty search, whether constrained or unconstrained. However, the best algorithm is contingent on the particularities of the search space and the genetic operators used. Additionally, the proposed enhancement of offspring boosting is shown to enhance performance in all cases of two-population novelty search.

Keywords

Genetic algorithms, novelty search, constrained optimization, two-population genetic algorithm, computer games, procedural content generation, level design.

1 Introduction

Genetic search has early on demonstrated its power at numerical optimization in a plethora of domains including evolutionary design (Lewis, 2008) and search-based procedural content generation (Togelius et al., 2011). When the search space is somehow constrained, however, genetic search has often faced significant challenges. The distinction between *feasible* and *infeasible* solutions is often unavoidable in engineering problems or during game design. For instance, a machine part must satisfy constraints on maximum cost or minimal performance, while a Sudoku puzzle must have a solution which satisfies its winning conditions. How best to handle infeasible individuals in a fitness-based stochastic search algorithm has become a topic of extensive research, and

has led to a variety of constrained optimization methods summarized by Michalewicz (1995b), Coello Coello (2010) and others. Despite the numerous solutions put forth, there has not yet been a consensus on a "general best" algorithm for handling constraints. Each technique has its benefits and drawbacks, making the choice of method dependent on the problem, the topology of the search space, the details of the implementation and, ultimately, personal preference.

Novelty search is a relatively new paradigm for stochastic search, put forth by Lehman and Stanley (2011a). Unlike traditional genetic search which performs selection according to a fitness, novelty search selects parents according to their divergence from a "norm". Individuals favored for selection are different — for a certain distance heuristic — from other individuals in the same population as well as from past novel discoveries. While novelty search has shown potential in robot control (Morse et al., 2013), maze navigation (Lehman and Stanley, 2008) and terrain design (Liapis et al., 2013f), there has been limited interest in exploring its performance in constrained spaces (Lehman and Stanley, 2010). However, domains such as robot control and game content generation come with several engineering and playability constraints respectively; this makes constrained novelty search a useful addition to current methods. Like objective-driven search before it, novelty search faces significant challenges when dealing with infeasible individuals. As it follows a different selection process, many of the methods used in constrained optimization, such as penalizing the fitness score of infeasible individuals, can not be directly applied.

This paper aims to introduce algorithms and methods within the area of constrained novelty search and explore how different parameters, genetic operators and search spaces affect their performance. The paper presents existing work on constrained novelty search, in the form of minimal criteria novelty search (Lehman and Stanley, 2010), and proposes two-population approaches which isolate infeasible individuals in a separate population evolving along with a population of feasible individuals. Two variants of two-population novelty search are proposed, applying novelty search either in the feasible population alone (with Feasible-Infeasible Novelty Search) or in both populations (with Feasible-Infeasible Dual Novelty Search). Constrained novelty search methods are tested on the domain of game level generation, as the need for diverse yet playable (i.e. feasible) game levels goes beyond academic interest and constitutes a commercial necessity for many game titles. Experiments in this domain demonstrate the differences in optimization behavior between approaches and indicate that, like objective-driven constrained optimization, the choice of method for constrained novelty search largely depends on the nature of the problem and its intended outcomes, as well as the topology of the search space and the genetic operators used.

This paper builds upon and extends the work of Liapis et al. (2013c), which introduced two-population novelty search methods and tested them on the domain of game level generation. To a large extent, the choice of game levels as a test domain originates in earlier experiments with Feasible-Infeasible Novelty Search on *Sentient Sketchbook* (Liapis et al., 2013d), a computer-aided design tool. The current paper expands on previous work by testing constrained novelty search methods on a new type of map along with the two maps of past experiments. Moreover, more rigorous experiments are performed, testing the impact of the genetic operators used, the size of the total population, the offspring boost addition to two-population algorithms and the injection of feasible individuals in the initial population. Finally, the paper proposes a number of domains where constrained novelty search can be valuable.

The structure of the paper is as follows: Section 2 provides an overview of the do-

mains of constrained optimization, novelty search and content generation, while Section 3 presents the two-population methods of constrained novelty search. Section 4 presents the domain of game level generation on which constrained novelty search will be tested, along with details of the genetic algorithm used, the constraints on playability and the characterization of distance between two levels. Section 5 puts forth a set of hypotheses on the superiority of two-population novelty search methods over existing approaches and tests them in different constrained spaces and experimental setups. Section 6 suggests extensions to the current experimental protocol as well as other domains for testing constrained novelty search; the paper concludes with Section 7.

2 Related Work

Novelty search is utilized in this paper for the constrained optimization of game content. A short survey of the relevant domains is presented below.

2.1 Constrained Optimization

While evolutionary algorithms have a rich history of successful applications in solving numerical optimization problems, it has never been straightforward how constraints should be handled. Such constraints are ubiquitous and often non-eliminable in e.g. engineering problems (Michalewicz et al., 1996), where solutions are often required to satisfy a minimal functional performance or safety and a maximal cost or size. Constraints divide the search space into feasible and infeasible spaces; depending on the problem, the feasible space may be fragmented, non-convex, or much smaller than than the infeasible space (see Fig. 2a). The greatest challenge in constrained search spaces is the issue of handling *infeasible individuals*, i.e. individuals which do not satisfy one or more of the constraints. This challenge is augmented by the fact that in many cases optimal feasible solutions lie in the border between feasible and infeasible space (Schoenauer and Michalewicz, 1996), such as an inexpensive engineering component with a risk of failure close to the safety threshold.

During the early period of constrained optimization research, infeasible individuals were simply assigned a fitness of zero; in most selection/replacement methods this would result in infeasible individuals being killed off in favor of feasible individuals. This *death penalty* to the infeasible individuals may not be particularly destructive in cases where infeasible individuals are rare; however, it has been argued against by Michalewicz (1995a), as substantial genetic information stored in infeasible individuals is lost. In highly constrained spaces, where a feasible individual does not exist in the initial population, genetic search incorporating the death penalty amounts to random search until the first feasible individual is discovered.

The most popular method for handling constraints in genetic optimization is to reduce the fitness score of infeasible individuals by a *penalty* score. Such penalties can be a constant value, a measure of feasibility or dynamically adapted according to the state of search. Different methods of penalizing the fitness function of infeasible individuals are surveyed by Coello Coello (2010). Designing penalty functions is not a trivial task, as penalties which are too high may amount to a death penalty while penalties which are too low may lead to superfluous search in the infeasible space.

Infeasible individuals can also be converted to feasible individuals through some procedure of *repair*. In many cases, the "cost" of repairing such an individual is applied to its fitness score as a penalty. The repair mechanism is useful in cases where evaluating an infeasible individual is not possible via the same fitness function used for feasible individuals. For instance, if the infeasible individual cannot be simulated

(e.g. for having negative mass with the physics model used) in a simulation-based evaluation process, converting the infeasible individual to feasible is the only way for calculating its fitness. However, designing a repair function requires significant domain knowledge and is often as challenging as solving the original problem, while applying a penalty based on repair costs has the same issues as other penalty functions.

Constrained optimization can also be carried out by evolving infeasible individuals separately from feasible ones. As infeasible individuals do not have to compete with feasible ones within their population, their evaluation can be decoupled from the feasible individuals' fitness function. While other approaches mate infeasible individuals with feasible ones (Kramer and Schwefel, 2006), the Feasible-Infeasible two population Genetic Algorithm (FI-2pop GA) only allows members of the same population to breed (Kimbrough et al., 2008). However, feasible offspring of infeasible parents migrate to the feasible population and vice versa; this indirect form of interbreeding allows the sharing of genetic information. While the feasible population evolves to maximize a problem-specific measure of quality, the infeasible population evolves to minimize its members' distance from feasibility (see Fig. 1a). By guiding infeasible individuals towards the feasible-infeasible border, FI-2pop increases the likelihood that their offspring will become both feasible and on the border of feasibility, where the optimal solution often lies (Schoenauer and Michalewicz, 1996).

2.2 Novelty Search

Novelty search has been proposed by Lehman and Stanley (2011a) as an alternative to objective-driven search, and has shown great potential in domains where a fitness function is deceptive, difficult to quantify, or subjective. Instead of evolving towards maximizing an objective function approximating the quality of a solution, novelty search evolves towards diversifying the solutions in a population. Novelty search selects an individual according to its *novelty score*, which is the average *distance* between the individual and its closest neighbors. This distance function is usually a characterization of the phenotype; for robots or maze-solving agents, the distance function becomes a behavioral characterization, comparing the two robots' positions sampled at one second intervals (Lehman and Stanley, 2011b) or the two agents' final positions (Lehman and Stanley, 2008), respectively. In order to prompt exploration of the search space, novelty search maintains a *novel archive*: in each generation, individuals with the highest novelty scores are added to the novel archive (in some cases, only if their novelty score is higher than a threshold). When calculating the novelty score, the individual's closest neighbors are drawn both from the current generation and from the novel archive (see Fig. 1b). As the novel archive expands with each generation, the novelty score of an area of the search space decreases; therefore, individuals in less explored areas of the search space are favored as they have fewer close neighbors in the novel archive.

The notion of *minimal criteria novelty search* (MCNS) was proposed by Lehman and Stanley (2010) in order to limit novelty search to the "useful" portions of the search space. MCNS applies the death penalty to infeasible individuals by reducing their novelty score to zero. Tested on navigating open mazes (without exterior walls), MCNS performed better than unconstrained novelty search as the latter explored areas outside of the maze which led the agent away from the exit. For highly constrained problems, however, Lehman and Stanley warn that a population of infeasible individuals will simply perform random search; as a solution, they suggest that a known feasible individual is inserted in the population to jump-start evolution. Even with such measures, however, the fact that all infeasible offspring of feasible parents are almost guaranteed

to be replaced results in losses of genetic information as argued by Michalewicz (1995a). Depending on the shape of the infeasible space, this may result in search being carried out on a small island of feasible space with no way of exploring further.

2.3 Procedural Generation of Game Content

Digital games often use *procedural content generation* (PCG) in order to speed up development as well as to present fresh and surprising experiences to players. The vast amount of assets needed in modern games drives companies to use computer-generated content either while developing a game or while the end-users are playing it. For content generated during development time (such as the terrain of a game's world), designers are safe in the knowledge that they can reject or edit any unwanted generated results; novelty and inspiration are more of a requirement from such generative algorithms than quality or playability. When game elements are generated while the end-user is playing, however, quality is imperative since a generated level that is too challenging or a generated 3D model with impossible geometry is detrimental to the player's engagement. Since the early days of Rogue (Toy and Wichman, 1980) and ELITE (Acornsoft, 1981), games have used procedurally generated content to provide a potentially endless experience to their players. Many modern computer games use carefully crafted algorithms to generate content such as weapons in Borderlands (Gearbox, 2009) or creatures in Spore (Maxis, 2008); however, the most popular use of *procedural content generation* (PCG) in commercial games remains game levels as in Torchlight 2 (Runic, 2012) or the gameworld in Minecraft (Mojang, 2011). Games which rely on generated content usually advertise the great replayability value they offer. For instance, best-selling PC game Diablo III (Blizzard, 2012) states that "[previous] games established the series' hallmarks: randomized levels, the relentless onslaught of monsters and events in a perpetually fresh world, [...][1].

The game industry therefore has a fundamental need for generators able to create both "good" and diverse content. Constrained novelty search is particularly suited for this task, as it targets diversity via the search for novelty while it ensures quality via the satisfaction of constraints. By collecting past content seen by the player (either generated or hand-crafted) into a novel archive, novelty search ensures that new content will be unlike those previously encountered. Constraining novelty search to playable — or sufficiently entertaining, challenging or balanced — game content ensures that players are not likely to experience unwanted content. Additionally, novelty search can be used to aid a game developer during creative tasks. Computer-aided design tools can benefit from computer generated content as alternatives to (Liapis et al., 2013d) or elaborations of (Liapis et al., 2013f) a user's current design; such suggestions can potentially speed up the design process and enhance the creativity of the human user (Yannakakis et al., 2014).

Academic interest in the procedural generation of game content is relatively new, but is considered one of the more promising directions of game AI research (Yannakakis, 2012). The *search-based procedural content generation* paradigm (Togelius et al., 2011) often uses genetic algorithms to search for new content which maximize a fitness function pertaining to its entertainment value (Yannakakis and Togelius, 2011), its learnability (Browne and Maire, 2010), its visual appeal (Liapis et al., 2012), its appropriateness to a specific user (Shaker et al., 2010) or multiple gameplay objectives (Togelius et al., 2013). Where fitness functions are less straightforward or subjective,

[1] From the official "What is Diablo 3?" page at Blizzard's website: *http://us.battle.net/d3/en/game/what-is* (Retrieved 14 February 2014)

content is generated via user interaction in the form of interactive evolution (Risi et al., 2012; Cardamone et al., 2011; Liapis et al., 2013a). Constraints on playability in search-based PCG have often been addressed by setting infeasible individuals' fitness to zero, although the FI-2pop GA paradigm has also been applied for generating playable platformer levels (Sorenson et al., 2011), strategy game levels (Liapis et al., 2013e) or spaceships (Liapis et al., 2011).

3 Two-Population Novelty Search Methods

As presented in Section 2.2, novelty search favors individuals with a high novelty score which is indicative of their divergence from current and past solutions. This novelty score $\rho(i)$, presented in Eq. (1), is the average distance between individual i and its k closest individuals, either in the current population or in a novel archive:

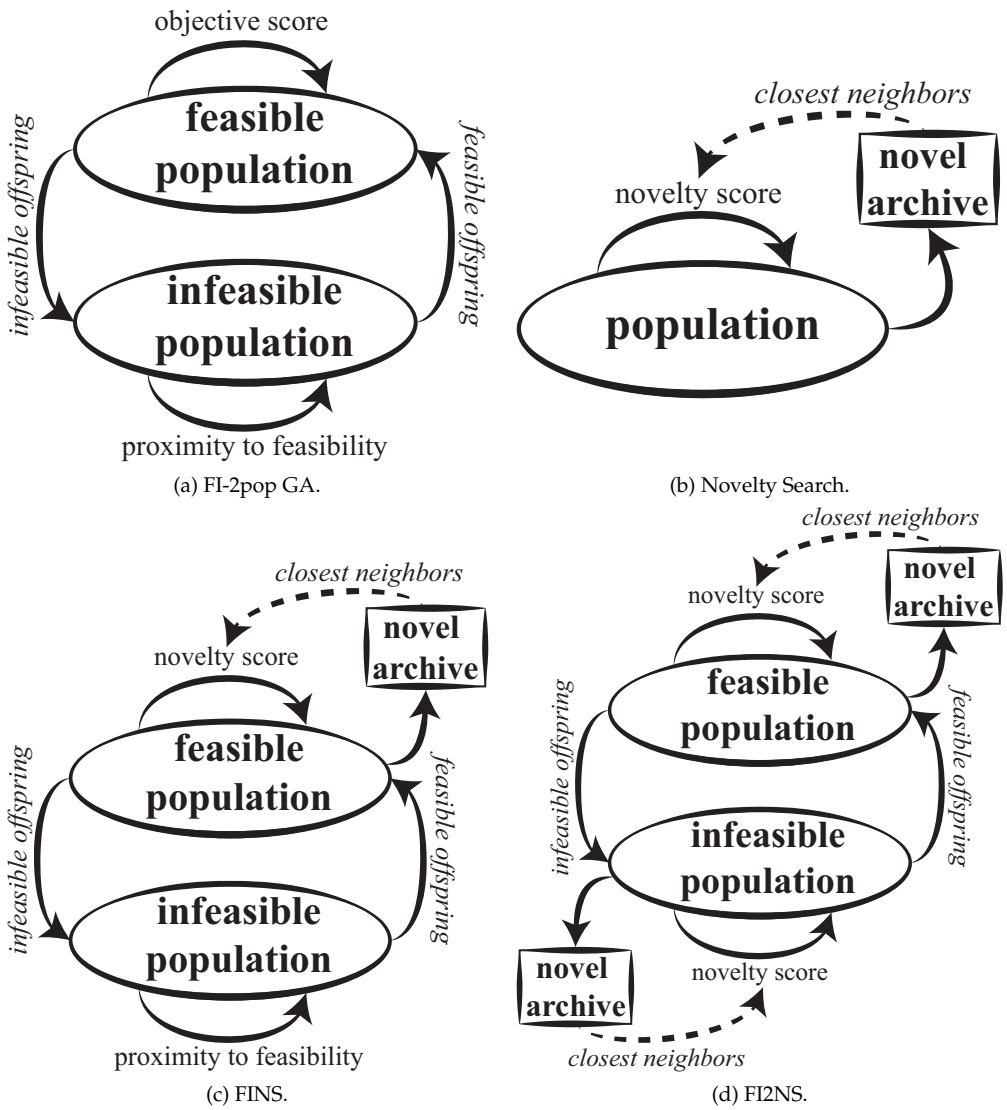
$$\rho(i) = \frac{1}{k}\sum_{j=1}^{k} d(i, \mu_j) \tag{1}$$

where μ_j is the j-th-nearest neighbor of i (within the population and in the archive of novel individuals); distance $d(i, j)$ is a domain-dependent metric which evaluates the "difference" between individuals i and j.

Although many prominent techniques for handling constraints with genetic algorithms penalize the fitness scores of infeasible individuals, applying penalties to novelty search is not straightforward considering the novelty score of Eq. (1). It is unclear, for instance, whether a penalty should be applied to $\rho(i)$ for infeasible i or to $d(i, j)$ for feasible i but infeasible j. It is therefore preferable to avoid comparisons between infeasible and feasible individuals. The FI-2pop GA maintains two populations so that infeasible individuals do not compete with feasible ones for the purposes of selection; feasible parents can thus be selected using a completely different criterion (e.g. novelty search) than infeasible ones. Additionally, feasible offspring of infeasible individuals migrate to the feasible population and increase its diversity, which coincides with the goals of novelty search among feasible individuals.

This paper presents two variations of this two-population approach, adapted to the purposes of novelty search. *Feasible-infeasible novelty search* (FINS) evolves feasible individuals, in their separate population, towards maximizing the novelty score $\rho(i)$ as per Equation (1) while infeasible individuals evolve towards minimizing a measure of their distance from feasibility f_{inf} (see Fig. 1c and Fig. 2b). In order to test the impact of the f_{inf} heuristic and do away with objective-driven optimization on both populations, the *feasible-infeasible dual novelty search* (FI2NS) performs novelty search on both the feasible and the infeasible population (see Fig. 2c). For FI2NS, novelty search is carried out independently in each population with two separate archives of feasible and infeasible novel individuals (see Fig. 1d); while both populations may use the same $\rho(i)$ metric, only the closest neighbors in the same population and archive are considered. Maintaining two populations for either FINS and FI2NS ensures that distances between feasible and infeasible individuals are not considered in the calculation of Eq. (1).

In the FI-2pop GA paradigm, the number of offspring for each population is equal to the current population's size. However, previous experiments in constrained optimization have indicated that an *offspring boost* on the feasible population was beneficial for enhancing the optimization behavior of feasible individuals. When the feasible population is smaller than the infeasible population, the offspring boost mechanism forces members of the feasible population to create a number of offspring equal to 50% of the



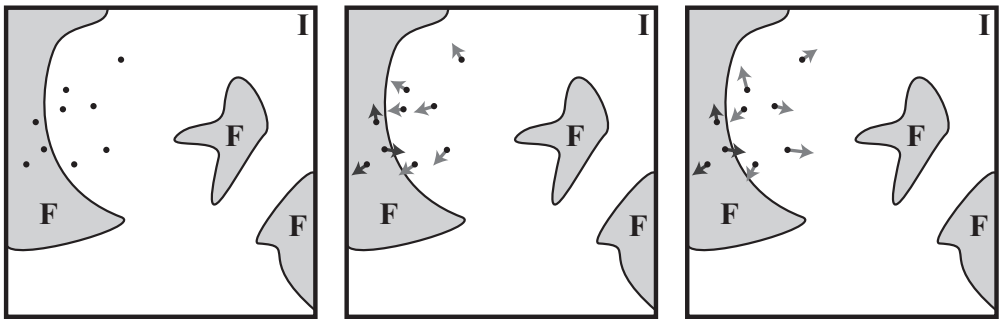
(a) FI-2pop GA.

(b) Novelty Search.

(c) FINS.

(d) FI2NS.

Figure 1: Diagrams of the two-population novelty search algorithms (Fig. 1c–1d), as well as their inspirations (Fig. 1a–1b). For FI-2pop GA, the feasible population evolves to maximize an objective score, while the infeasible population evolves to minimize its members' distance from feasibility. For Novelty Search, a single population evolves to maximize a novelty score which measures the average distance of an individual from its closest neighbors in the current population and in a *novel archive* which is updated in every generation with the most novel individuals. For FINS, the feasible population performs novelty search using a novel archive containing only feasible individuals, while the infeasible population evolves to minimize the distance from feasibility. For FI2NS, the feasible population performs novelty search using a novel archive containing only feasible individuals; the infeasible population performs novelty search as well, but uses its own novel archive with only infeasible individuals and calculates the novelty score by measuring distance only between infeasible individuals.



(a) Feasible and Infeasible Spaces. (b) FINS search process. (c) FI2NS search process.

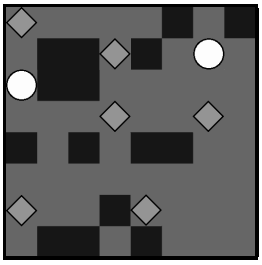
Figure 2: A visualization of the search process for two-population novelty search methods. Fig. 2a shows a possible search space divided into infeasible space (I) and islands of non-convex feasible space (F), along with a randomly initialized population (black dots). With FINS, the infeasible individuals evolve to minimize their infeasibility, ideally approaching the closest border with feasibility (Fig. 2b). With FI2NS, the infeasible individuals evolve to maximize their diversity, ideally discovering faraway islands of feasible space (Fig. 2c). For both FINS and FI2NS, feasible individuals evolve towards maximizing their novelty; in the example shown, novelty search is likely to cause feasible individuals on the feasibility border to become infeasible.

total size of the two populations. The number of offspring in the infeasible population is reduced accordingly to keep the total population size steady. Unless otherwise noted, all experiments in this paper apply the proposed offspring boost enhancement to FINS and FI2NS methods. The impact of the offspring boost will also be tested in experiments of Section 5, with runs of FINS and FI2NS with and without the boost.

4 Game Level Creation

Procedurally generated game levels — as the most popular type of generated game content in the game industry as well as within academia — have a need for both playability and diversity, which makes constrained novelty search an obvious solution. For the purposes of this paper, constrained novelty search will be used to evolve strategy game levels. Section 6.2 will elaborate on other domains where constrained novelty search can also be useful.

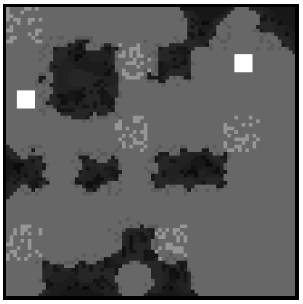
The generation of strategy game levels via constrained novelty search is motivated by the Sentient Sketchbook design tool (Liapis et al., 2013d), which allows human designers to draw their own maps while an artificial designer procedurally generates alternatives to those, in real-time, and presents them to the users. The maps generated by Sentient Sketchbook are represented as *map sketches*. Such sketches are minimal abstractions of game levels, which contain the bare essentials for strategic gameplay; map sketches, however, can be transformed via rule-based processes with some stochasticity into elaborate, complete levels (see Fig. 3). Although many different types of game levels (including dungeons and first-person shooter maps) can be represented as map sketches (Liapis et al., 2013g), this experiment will focus on strategy game levels: therefore, map sketches contain passable tiles, impassable tiles, player bases and resources. As in a typical strategy game, each player is assumed to start the game at one of the player bases, from where they must collect resources in order to build units; units travel



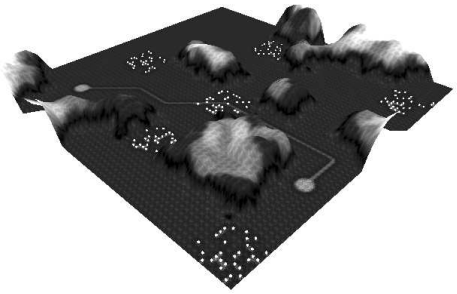
(a) Generated map sketch.

1	0	0	0	0	4	0	4
0	4	4	1	4	0	2	0
2	4	4	0	0	0	0	0
0	0	0	1	0	0	1	0
4	0	4	0	4	4	0	0
0	0	0	0	0	0	0	0
1	0	0	4	1	0	0	0
0	4	4	0	4	0	0	0

(b) Genotype of the sketch in Fig. 3a.



(c) Detailed strategy game level constructed from the sketch in Fig. 3a.



(d) A strategy level in 3D, constructed from the sketch in Fig. 3a.

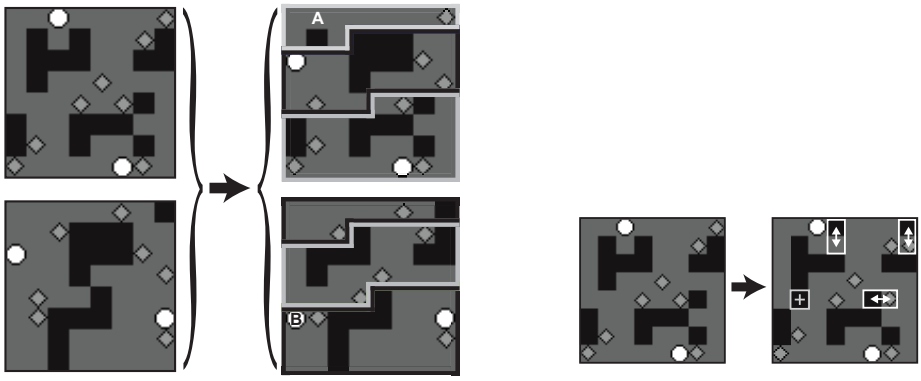
Figure 3: A map sketch generated by the methodology in Section 4 (Fig. 3a); passable tiles are shown in orange, impassable tiles in brown, player bases as circles and resources as rhombi; the genotype is an integer array with each tile stored as an integer (Fig. 3b). The map sketch can be transformed, via cellular automata, into a complete strategy game map in 2D (Fig. 3c) or in 3D (Fig. 3d); the detailed maps maintain the same navigational and playability properties of the sketch they are based on.

through passable tiles to attack and destroy the enemy players' bases.

4.1 Representation and Genetic Algorithm

The most significant benefit of the low-resolution map sketches is their ability to be mapped directly into a genotype. The genotype is an array of integers equal to the number of tiles of the map sketch: each integer thus determines the contents of a single map tile (see Fig. 3b).

In the experiments presented in this paper, the genotype is evolved towards maximizing a measure of diversity, or in the case of FINS towards minimizing the distance from feasibility. Evolution is carried out via fitness-proportionate roulette-wheel selection; the same parent may be selected more than once and thus generate multiple offspring, which is necessary when the number of offspring is different than the number of parents (e.g. when applying the offspring boost). In each population, the best individual is carried to the next generation while remaining individuals are replaced by new ones. Elitism does not seem valuable to novelty search, since the fittest individual in one generation is likely to have a low fitness in the next generation as it will also be present in the novel archive. However, the "minimal" elitism used ensures the presence of at least one feasible individual in each generation for two-population novelty search



(a) Example of 2-point crossover.

(b) Example of mutation.

Figure 4: Examples of genetic operators applied on game levels. Due to the direct representation, crossover creates two offspring with parts of the first map (light outline) and the second map (dark outline). Crossover may result in offspring with more bases or resources than their parents; assuming a designer constraint for maps with exactly two bases, the repair mechanism must be applied as the offspring have 3 bases and 1 base respectively. For the top map the repair mechanism removes a base chosen randomly between the three, and the base at location A is changed to passable tile. For the bottom map the repair mechanism adds a base on a randomly chosen passable tile, and the tile at location B is changed to base. Mutation changes a few tiles of the map: in Fig. 4b that amounts to 6% of map tiles. Mutated tiles may be shifted with an adjacent one (white arrows) or transformed from passable to impassable or vice versa (+ sign).

approaches. New individuals can be created via a 2-point crossover of two parents (see Fig. 4a) or via mutation (see Fig. 4b). Mutation alters between 5% to 20% of the maps' tiles (determined randomly): each tile has an equal chance of being swapped with a randomly chosen adjacent one, or transformed from passable to impassable and vice versa (bases and resources are not transformed). For populations evolving via novelty search, the 20 closest individuals are considered when calculating the novelty score $\rho(i)$ of Eq. 1 (i.e. $k = 20$), while the 5 highest scoring individuals of each generation are added to the novel archive.

4.2 Constraints and Feasibility

In the PCG domain, constraints on what is an acceptable game level can be envisioned as a minimal projected entertainment value or a minimal fairness in how contesting players are treated. In order to avoid elaborate and domain-specific constraint formulations which may limit the feasible space considerably, this paper will only consider the minimal criteria of playability. For strategy games, two minimal criteria exist: (a) there must be a minimum and a maximum number of bases and resources for a level to be playable (for instance, a strategy game requires two players and therefore at least two bases), and (b) all bases must be able to reach, via a passable path, all other bases and resources to enable a player to attack opponents and collect resources, respectively.

The number of bases and resources allowed is ultimately a designer decision, but

is also dependent on the size of the map: small maps can only have a few resources and even fewer bases, while larger maps might have sufficient open areas for a larger number of bases and therefore players. The current mutation scheme does not change the number of bases or resources on the map, and thus mutating a gene with a feasible number of bases and resources is guaranteed to generate a gene with a feasible number of bases and resources. When applying crossover, however, there is a likelihood that the offspring maps will have more or less bases and resources than their individual parents; in such cases, a *repair* mechanism alters the offspring's genotype to satisfy constraints on the number of bases and resources. The repair mechanism adds missing bases or resources to infeasible individuals with missing bases or resources respectively, or removes extraneous bases or resources from infeasible individuals with excess bases or resources respectively — without any penalties to their fitness. The process is stochastic, as missing bases or resources are added to randomly selected passable tiles while excess bases and resources are chosen at random and removed (see Fig. 4a). Although this repair mechanism increases the unpredictability of the crossover operator, it is not as destructive as creating two infeasible offspring.

When a base is not connected to another base or resource, there is no straightforward way of repairing the map; therefore, the connectivity of bases and resources is the hard constraint on which constrained novelty search methods will be tested. Feasible individuals have bases connected via passable paths with all other bases and resources; infeasible individuals have at least one disconnected path. For the purposes of FINS, the distance from feasibility is calculated by f_{inf} in Eq. 2. Infeasible individuals have $f_{inf} > 0$, while feasible individuals have $f_{inf} = 0$; this equality constraint is used in all constrained novelty search methods when testing for feasibility.

$$f_{inf} = \frac{u_b}{B(B-1)} + \frac{u_r}{RB} \tag{2}$$

where B and R is the number of bases and resources respectively; u_b and u_r is the number of disconnected paths between pairs of bases and between pairs of bases and resources respectively.

4.3 Distance Characterization

Measuring the difference between two maps is no straightforward task and the choice of a distance metric for novelty search is likely to affect the appearance of generated individuals. Since users of the Sentient Sketchbook tool inspect the generated map suggestions visually, it is assumed for the purposes of this study that visual diversity between generated maps is the most desirable quality. The distance metric for *visual diversity*, shown as d_{vis} in Eq. 3 compares two maps on a tile-by-tile basis: the fewer matching tiles between maps, the higher the score of d_{vis}. This d_{vis} metric will be used to calculate the novelty score according to Eq. 1 for all experiments in this paper.

$$d_{vis}(i,j) = \frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} D_{x,y}(i,j) \tag{3}$$

where W and H is the width and height of the map respectively and $D_{x,y}(i,j) = 0$ if at coordinates (x,y) the tile of map i is of the same type as that of map j and $D_{x,y}(i,j) = 1$ if it is of a different type.

5 Experiments

In order to assess the behavior of constrained novelty search methods, a number of experiments were conducted and reported below; these experiments compare between minimal criteria novelty search (MCNS), feasible-infeasible novelty search (FINS), feasible-infeasible dual novelty search (FI2NS) and traditional, unconstrained novelty search (NS). The introduction of two-population novelty search aims to test three hypotheses (*H1*, *H2*, *H3*) regarding existing methods; a fourth hypothesis (*H4*) is made for the addition of the offspring boost mechanism to the two-population methods:

H1: FINS can find feasible individuals faster than FI2NS, NS and MCNS in highly constrained search spaces, where a feasible individual is hard to discover and not likely to exist in the initial population. In such search spaces MCNS performs random search and both NS and FI2NS search for novelty in infeasible space, while the infeasible population of FINS is guided by a measure of feasibility.

H2: FINS and FI2NS can find more feasible individuals than NS. As NS does not distinguish between feasible and infeasible individuals, it is likely to explore infeasible space when the feasible space is small; this amounts to a few feasible individuals, limiting the efficiency of novelty search in that space.

H3: MCNS has lower diversity than all other methods, as it kills all infeasible individuals once a feasible individual is found. FINS, FI2NS and NS all retain infeasible individuals; such individuals are likely to eventually create feasible offspring and enhance the diversity of feasible individuals. On the other hand, MCNS kills infeasible offspring of feasible parents, thus forgetting their genetic information; this is likely to limit the effectiveness of novelty search near the border of feasible space and thus lower the diversity of feasible individuals.

H4: FINS and FI2NS become more efficient from the offspring boost mechanism in terms of the number and diversity of feasible individuals discovered. The feasible population of FINS and FI2NS must be large enough for search to be efficient; without the offspring boost, their feasible population may be too small.

To validate the above hypotheses, we derive four performance metrics (two for *H1* and one for each of *H2*, *H3*) to be used in the experiments of this paper: (*H1*a) the number of evolutionary runs (out of 50) where at least one feasible individual is discovered (n); (*H1*b) the first generation in which a feasible individual is discovered (g); (*H2*) the number of feasible individuals (p); and (*H3*) the diversity of feasible individuals ($\bar{d}_{vis}$). *H4* can be tested via p and $\bar{d}_{vis}$, since its goal is higher diversity afforded by larger feasible populations. Diversity is measured by calculating the average distance (d_{vis}) between all feasible individuals in the population (ignoring the novel archive). The values of p and $\bar{d}_{vis}$ are calculated after 100 generations, at which point the evolutionary run is terminated. Apart from n, performance metrics are averaged across 50 runs, with the standard deviation shown in parentheses. Significance throughout this paper is $\alpha = 5\%$, measured via standard t-tests; when comparing more than one method, the Bonferroni correction is applied (Dunn, 2012). Unless otherwise stated, experiments in this paper are performed on a total population of 100 individuals, which can be feasible or infeasible.

To showcase the algorithms' behavior in different constrained search spaces, experiments will be performed on small, medium and large maps. Small maps have 64 tiles (laid on a grid of 8 by 8 tiles), and must contain 2 bases and between 4 and 10

resources. The few bases and resources and small size makes the creation of feasible individuals likely, even by random generation: from 10^6 randomly generated maps, $3{\cdot}10^4$ are feasible. Large maps have 256 tiles (laid on a grid of 16 by 16 tiles), and must contain between 2 and 10 bases and between 4 and 30 resources. The large map size increases the likelihood that paths between bases and resources will be blocked, especially when either bases or resources are numerous; from 10^6 randomly generated large maps, only 10 are feasible. Medium maps have 144 tiles (laid on a grid of 12 by 12 tiles), and must contain 4 bases and between 8 and 20 resources. The map size should place medium maps somewhere between small and large maps in terms of ease of satisfying constraints, although medium maps have a minimum of 12 tiles that must be connected (bases and resources) versus the minimum of 6 tiles for small and large maps: from 10^6 randomly generated medium maps, 852 are feasible.

The algorithms behave differently depending on the search space and the genetic operators used. In order to test that, experiments documented in Section 5.1 compare the two-population approaches with unconstrained NS and MCNS and test the impact of the offspring boost mechanism. These experiments take place on a total population of 100 individuals, and start from a random initial population; to test how these parameters affect the behavior of the different methods, Section 5.2 tests population sizes both larger and smaller than 100 individuals, while Section 5.3 bypasses the challenge of discovering feasible individuals by injecting one or more feasible solutions into the initial population.

5.1 Impact of Genetic Operators

The choice of genetic operators can greatly affect the behavior of any genetic algorithm, even more so novelty search which relies on divergence to guide it. This section will evaluate the behavior of the four novelty search methods (three constrained and one unconstrained) in three different map sizes. All experiments will compare how the different methods behave when the genetic algorithm uses 2-point crossover (with a 1% chance of mutation) and when it uses only mutation; as a shorthand, the former operators will be identified as *recombination* and the latter as *mutation*. Unless otherwise noted, FINS and FI2NS use the offspring boost mechanism; experiments in the second part of this section validate the *H4* hypothesis and support the use of the offspring boost in all experiments in this paper.

Table 1 shows the performance metrics of different methods of novelty search evolving via the 2-point crossover of two parents and a mutation chance of 1% on the offspring. Comparing between the different map sizes, it is immediately obvious that the performance of different methods is greatly affected by the shape of the feasible space: in highly constrained spaces, minimal criteria novelty search (MCNS) and unconstrained novelty search (NS) appear to struggle to discover or retain feasible individuals, respectively. MCNS discovers a feasible individual in 43 out of 50 runs for medium maps and only in 7 out of 50 runs for large maps; as the initial population does not contain feasible individuals, MCNS sets the fitness of the entire population to 0 and performs random search until a feasible individual is found. For experiments with recombination, it is obvious that random search is not particularly efficient at finding parents whose offspring are feasible. Unconstrained novelty search, on the other hand, discovers feasible individuals more often than MCNS, but does not succeed at retaining them in the population: it was not uncommon for runs with NS to discover a single feasible individual in one generation, only to have it disappear in the next. As NS does not have an explicit (via a feasibility check) or implicit (via the distance metric)

Map Size	Method	n	g	p	$\bar{d}_{vis}$
Small	NS	50	0.0 (0.2)	**14.7 (6.3)**	0.523 (0.079)
	MCNS	50	0.1 (0.4)	**96.7 (2.1)**	**0.402 (0.019)**
	FINS	50	0.0 (0.1)	**51.6 (5.4)**	**0.467 (0.016)**
	FI2NS	50	0.1 (0.4)	**48.9 (3.7)**	0.523 (0.010)
Medium	NS	50	7.3 (7.1)	**1.2 (1.4)**	0.192 (0.252)
	MCNS	43	13.4 (12.5)	**94.7 (2.9)**	0.299 (0.030)
	FINS	50	**2.6 (1.6)**	**50.1 (5.1)**	**0.418 (0.016)**
	FI2NS	50	8.7 (8.6)	**47.5 (3.6)**	**0.481 (0.012)**
Large	NS	34	53.7 (25.0)	**0.1 (0.3)**	**0.000 (0.000)**
	MCNS	7	43.3 (27.6)	**85.9 (28.2)**	**0.176 (0.077)**
	FINS	50	8.5 (3.9)	**48.9 (4.2)**	**0.361 (0.023)**
	FI2NS	30	60.5 (20.9)	**44.1 (4.5)**	**0.293 (0.119)**

Table 1: Performance metrics for different novelty search approaches used with 2-point crossover and 1% chance of mutation: number of runs (out of 50) where a feasible individual was discovered (n) and the first generation in which it occurred (g), as well as the final number of feasible individuals (p) and their average diversity ($\bar{d}_{vis}$) of each experiment. For metrics other than n, results are averaged across 50 runs with standard deviation in parentheses; significantly different values from all other methods for the same map size are displayed in bold, while (significantly) best values for each metric are underlined. Note that the best value is the lowest for g and the highest for p and $\bar{d}_{vis}$.

indication that the individual discovered is valuable, it was often discarded in favor of more visually different yet infeasible individuals. Both FINS and FI2NS have an explicit indication of feasible individuals' value, and once discovered these individuals are prompted to generate numerous offspring — equal to 50% of the total population due to the offspring boost mechanism. Unlike the random search of MCNS and the novelty search of FI2NS and NS, FINS attempts to minimize infeasible individuals' distance from feasibility, and this objective-driven search leads to the discovery of feasible individuals faster and more reliably (based on the low deviation of g) than the other methods in medium and large maps. Regarding the number of final feasible individuals, MCNS unsurprisingly has far more feasible individuals than other methods in all runs where a feasible individual was discovered, for all map sizes. Likely due to the fact that all infeasible individuals are killed, however, MCNS has a significantly lower diversity in the feasible population than all methods (excluding NS in medium or large maps which hardly contains any feasible individuals). In less constrained spaces such as with small maps, NS can discover a sufficient number of feasible individuals to measure their diversity; its final diversity is significantly higher than that of MCNS and FINS. For medium maps FI2NS achieves a significantly higher diversity than all other approaches, likely due to the fact that novelty search in the infeasible population discovers feasible individuals in distant or segmented areas of the search space. For large maps, the feasible population of FINS is significantly more diverse than other methods', likely due to the fact that FINS discovers individuals much earlier than other methods and thus performs novelty search on the feasible population for more generations.

Table 2 shows the performance metrics of different methods of novelty search evolving via the mutation of a single parent. Comparing the performance metrics with those of Table 1, it is clear that mutation, as a whole, tends to cause different novelty search methods to exhibit a similar behavior. Feasible individuals are discovered in

Map Size	Method	n	g	p	$\bar{d}_{vis}$
Small	NS	50	0.0 (0.1)	**10.9 (4.0)**	0.599 (0.032)
	MCNS	50	0.1 (0.2)	**77.6 (6.3)**	**0.577 (0.017)**
	FINS	50	0.0 (0.2)	38.9 (4.5)	0.603 (0.007)
	FI2NS	50	0.0 (0.0)	38.5 (4.3)	0.607 (0.010)
Medium	NS	50	4.0 (2.8)	**1.7 (1.5)**	**0.392 (0.282)**
	MCNS	50	4.7 (2.9)	**55.2 (7.7)**	0.567 (0.011)
	FINS	50	**2.5 (1.6)**	**28.2 (4.3)**	0.577 (0.012)
	FI2NS	50	4.0 (2.8)	**25.4 (4.7)**	0.572 (0.015)
Large	NS	50	16.9 (9.0)	**0.2 (0.4)**	**0.000 (0.000)**
	MCNS	50	20.4 (11.4)	**31.8 (6.0)**	0.540 (0.018)
	FINS	50	**6.2 (1.9)**	16.1 (4.2)	0.544 (0.026)
	FI2NS	50	17.1 (7.8)	15.2 (4.2)	**0.523 (0.040)**

Table 2: Performance metrics for different novelty search approaches used only with mutation: number of runs (out of 50) where a feasible individual was discovered (n) and the first generation in which it occurred (g), as well as the final number of feasible individuals (p) and their average diversity ($\bar{d}_{vis}$) of each experiment. For metrics other than n, results are averaged across 50 runs with standard deviation in parentheses; significantly different values from all other methods for the same map size are displayed in bold, while (significantly) best values for each metric are underlined. Note that the best value is the lowest for g and the highest for p and $\bar{d}_{vis}$.

all runs, even in large maps, and the discrepancies in g and $\bar{d}_{vis}$ between methods are largely subdued. All behaviors (excluding NS in medium and large maps) reach similar values of final diversity, which are significantly higher than the values obtained by the same methods using recombination. This should not be surprising given the very nature of recombination as well as the tile-based distance metric used to measure diversity. Given an appropriate "geometric" recombination operator, evolutionary search can be seen a convex search as it searches inside the hyper-volume specified by the individuals in the initial population (Moraglio, 2011). In the current problem, the direct genotype to phenotype mapping makes it probable that the recombination operator is geometrical or almost geometrical; this could be a good explanation for the low observed diversity in experiments with prevalent recombination. More specifically, an individual created via crossover inherits parts of the map from either parent, resulting in a low distance d_{vis} between the offspring and either parent (see Fig. 4a). On the other hand, a map generated via mutation inherits parts of the map from a single parent, while the random mutation means that offspring of the same parent may still be dissimilar from each other (see Fig. 4b). Regarding the number of feasible individuals, however, it is surprising that all methods have fewer feasible individuals with mutation than with recombination, regardless of map size. From the results, it is obvious that applying mutation to a feasible individual is more likely to render it infeasible than when recombining it with another individual (usually feasible); this is likely due to the fact that mutation can add impassable tiles to a map, which can block a previously passable path. However, this same mechanism of mutation is also more likely to discover feasible individuals; even the random search of MCNS discovers large feasible maps in all runs much faster than with recombination, although FINS still manages to discover such maps faster and more reliably.

Results in this section point to very different behaviors in all performance metrics between experiments with mutation and experiments with recombination. To ascertain

		Recombination		Mutation	
Map Size	Method	p	$\bar{d}_{vis}$	p	$\bar{d}_{vis}$
Small	FINS$_{nb}$	37.8 (10.6)	0.426 (0.025)	19.6 (5.6)	0.597 (0.013)
	FINS	51.6 (5.4)	0.467 (0.016)	38.9 (4.5)	0.603 (0.007)
	FI2NS$_{nb}$	20.4 (7.3)	0.511 (0.024)	12.4 (3.7)	0.605 (0.018)
	FI2NS	48.9 (3.7)	0.523 (0.010)	38.5 (4.3)	0.607 (0.010)
Medium	FINS$_{nb}$	28.1 (9.7)	0.337 (0.038)	6.4 (3.0)	0.556 (0.086)
	FINS	50.1 (5.1)	0.418 (0.016)	28.2 (4.3)	0.577 (0.012)
	FI2NS$_{nb}$	3.5 (2.6)	0.288 (0.195)	2.9 (1.2)	0.495 (0.197)
	FI2NS	47.5 (3.6)	0.481 (0.012)	25.4 (4.7)	0.572 (0.015)
Large	FINS$_{nb}$	23.4 (12.3)	0.239 (0.087)	2.8 (2.0)	0.410 (0.251)
	FINS	48.9 (4.2)	0.361 (0.023)	16.1 (4.2)	0.544 (0.026)
	FI2NS$_{nb}$	1.2 (0.7)	0.036 (0.128)	1.2 (0.5)	0.096 (0.218)
	FI2NS	44.1 (4.5)	0.293 (0.119)	15.2 (4.2)	0.523 (0.040)

Table 3: Performance metrics with and without (shown as $_{nb}$) the offspring boost mechanism, for FINS and FI2NS evolving via 2-point crossover and 1% chance of mutation (Recombination column) and only mutation (Mutation column). Since the offspring boost takes effect only after a feasible individual is discovered, n and g are not expected to differ from those in Tables 1 and 2 and are omitted. p and $\bar{d}_{vis}$ are compared between the same method (FINS or FI2NS) with and without the offspring boost; significantly better (higher) values are underlined. Results are averaged across 50 runs, with standard deviation in parentheses.

whether such differences are persistent across population sizes or initial populations, remaining experiments in this paper will also be testing both sets of genetic operators.

Impact of the Offspring Boost

Section 3 proposed an addition to the two-population novelty search methods for increasing the size of the feasible population. This mechanism, identified as *offspring boost*, increases the number of offspring created by the feasible population to a minimum of 50% of the total population or the current size of the feasible population, whichever is greater. The hypothesis *H4* for this addition is that the feasible population of FINS and FI2NS must be large enough for search to be efficient. To test *H4*, 50 runs of FINS and FI2NS without the offspring boost (denoted as FINS$_{nb}$ and FI2NS$_{nb}$ respectively) are compared with the 50 runs of FINS and FI2NS from Tables 1 and 2, which use the offspring boost. The results of these tests are shown in Table 3.

As expected, the offspring boost results in a significantly larger feasible population, since offspring of feasible parents are more likely to be feasible than offspring of infeasible parents. FI2NS benefits from the boost the most: without the boost, its final population is small for small maps and for medium or large maps it often consists of a single feasible individual. FI2NS$_{nb}$ does not have a much larger number of feasible individuals than NS, although their difference is still significant. While FINS using recombination often has a sizable feasible population even without the boost, when using mutation FINS similarly relies on the boost to reach an adequate population size. The larger feasible populations afforded by the boost allow both FINS and FI2NS to search for novelty more efficiently in the feasible space, usually resulting in significantly higher diversity values than the same methods without the boost. From the results of Table 3, the first part of hypothesis *H4* that the offspring boost results in more feasible individuals is validated in 12 out of 12 cases, regardless of map size, genetic operators, and two-population novelty search method. The second part of hypothesis

H4 that the offspring boost results in more diverse feasible individuals is validated for FINS in 5 out of 6 cases, and similarly for FI2NS in 5 out of 6 cases. Since *H4* is validated for the overwhelming majority of cases, every FINS and FI2NS method uses the offspring boost in the remaining experiments.

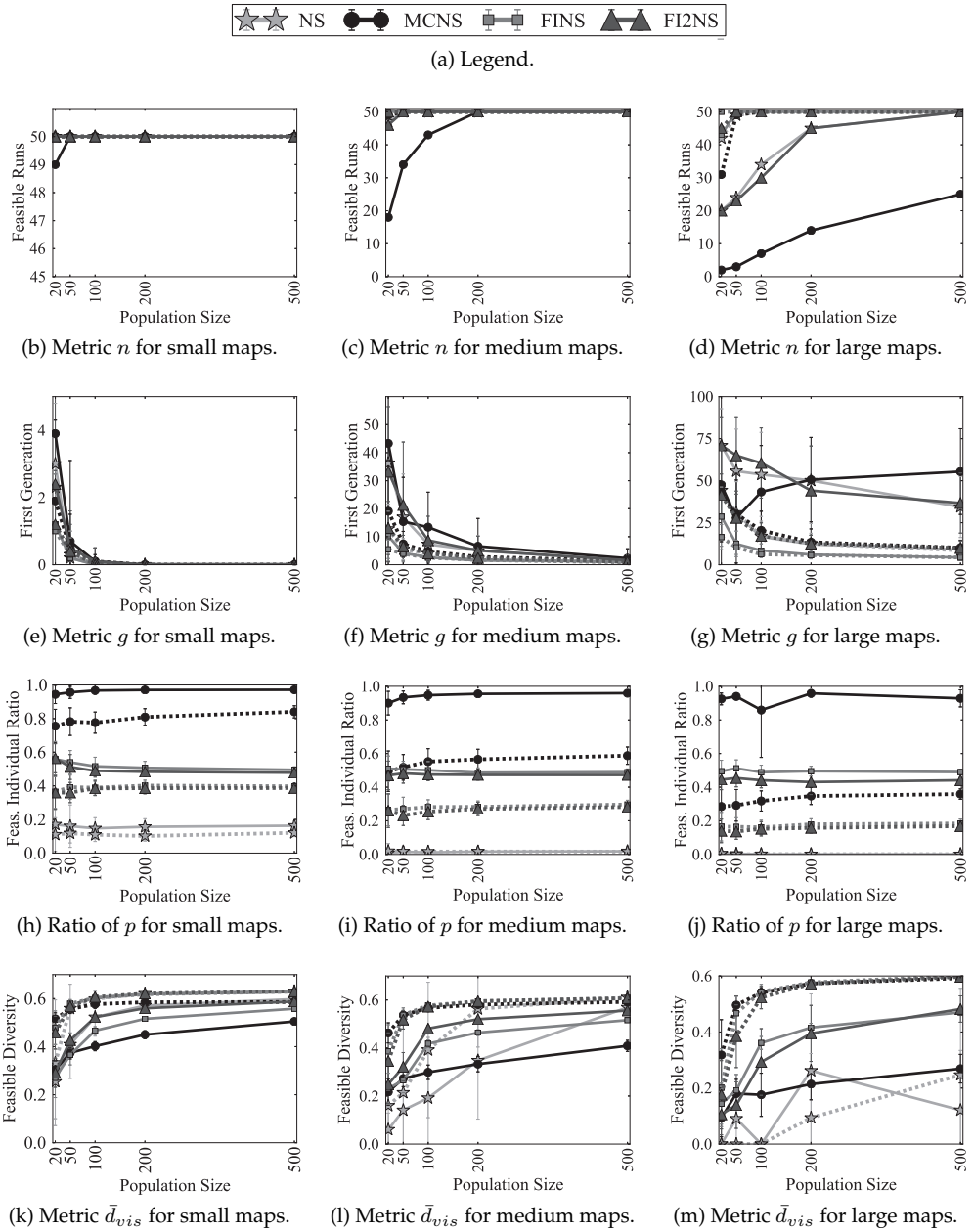
5.2 Impact of Population Size

Since most genetic algorithms benefit from larger populations for the additional parallel search that they afford, the same should hold true for constrained novelty search. A larger population size is more likely to create, even by chance, maps that are more different from each other and thus better steer novelty search towards higher diversity. Moreover, a larger population is more likely to discover a feasible individual in highly constrained spaces, regardless of the type of search.

While other experiments in this paper de facto use a total population of 100 individuals, which includes both feasible and infeasible ones, Figure 5 displays how different performance metrics change with smaller or larger populations. Five different population sizes are tested: 20, 50, 100, 200 and 500 individuals. Several interesting conclusions can be drawn from Fig. 5, although they largely depend on the map size and the genetic operators used.

For all map sizes, the number of feasible runs increases as the size of the total population increases. This is hardly surprising since a larger population of infeasible individuals is faster at reaching the border of feasibility for FINS or exploring the infeasible space for FI2NS and NS. As MCNS lacks an informed heuristic for searching infeasible space, it benefits the most from the parallel random search afforded by large population sizes. The discovery of the first feasible individual is also considerably faster with larger population sizes (such as 200 or 500 individuals) for all methods, although FINS is the least affected. On the other hand, the ratio of final feasible individuals to the total population does not seem to be particularly sensitive to different population sizes. The average diversity of the final feasible individuals increases with larger populations of 200 or 500, which is hardly surprising since most forms of genetic search benefit from larger population sizes; it is interesting to note, however, that recombination benefits more from large population sizes than mutation.

Small populations such as 20 or 50 individuals are quite detrimental for most methods. For small maps the discovery of feasible individuals is still fast and easy even for a population of 20, although the final feasible diversity is significantly lower compared to that for a population of 100. For medium maps, small populations do not particularly affect the discovery of feasible individuals for FINS, since it still consistently finds feasible individuals within the first 10 generations even for populations of 20. It does however affect the discovery of feasible individuals for FI2NS and NS, which discover individuals much later than in populations of 100, and especially MCNS which does not discover any feasible individuals in numerous runs when applying recombination. As with small maps, all methods fail to reach high values of feasible diversity for medium maps when evolving 20 or 50 individuals. For large maps, small populations are even more problematic, as even FINS struggles to discover feasible individuals with populations of 20. Other methods fare even worse, as MCNS, FI2NS and NS rarely discover feasible individuals with populations of 20; in fact, MCNS discovers a feasible large map only once when using recombination with populations of 20. Although it discovers feasible individuals rarely, MCNS can search for novelty in the feasible population more efficiently than other methods in very small populations; as other methods do not have a sufficiently large population in order to effectively explore



(a) Legend.

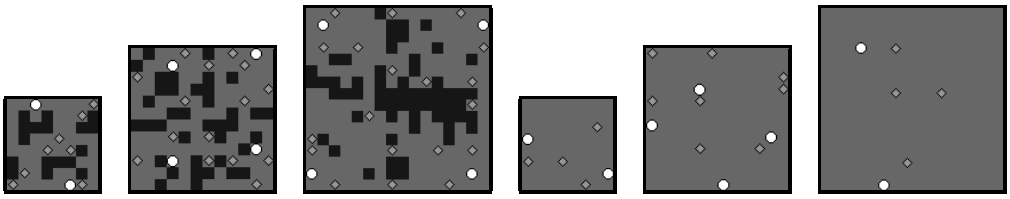
(b) Metric n for small maps. (c) Metric n for medium maps. (d) Metric n for large maps.

(e) Metric g for small maps. (f) Metric g for medium maps. (g) Metric g for large maps.

(h) Ratio of p for small maps. (i) Ratio of p for medium maps. (j) Ratio of p for large maps.

(k) Metric $\bar{d}_{vis}$ for small maps. (l) Metric $\bar{d}_{vis}$ for medium maps. (m) Metric $\bar{d}_{vis}$ for large maps.

Figure 5: Comparisons of the performance metrics between different population sizes. The ratio of feasible individuals p is normalized to the population size used (a ratio of 1.0 represents the entire population size). Solid lines show experiments with recombination while dotted lines show experiments with mutation. Results are averaged across 50 runs, with errorbars denoting standard deviation.



(a) Small $seed_{one}$ (b) Medium $seed_{one}$ (c) Large $seed_{one}$ (d) Small sample $seed_{all}$ (e) Medium sample $seed_{all}$ (f) Large sample $seed_{all}$

Figure 6: Types of initial seeds. Fig. 6a–6c show the feasible hand-crafted maps used to seed the initial population of the $seed_{one}$ experiments. Fig. 6d–6f show sample randomly generated feasible maps, with no impassable tiles, used to seed the initial population of the $seed_{all}$ experiments.

the feasible space, MCNS achieves significantly higher diversity scores in populations of 20 for experiments with mutation.

The differences found in Section 5.1 between experiments with recombination and experiments with mutation seem to be persistent for the different population sizes tested. The number of runs with no feasible individuals is much larger for experiments with recombination than for experiments with mutation, especially with smaller population sizes. Similarly, experiments with mutation discover feasible individuals earlier, and the differences become even more pronounced with larger population sizes. Experiments with mutation consistently have fewer final feasible individuals than experiments with recombination, for populations both small and large. Regarding diversity, while experiments with mutation and recombination both benefit from larger population sizes, recombination struggles much more with small populations than mutation. Both experiments with mutation and experiments with recombination have a significantly lower final diversity with a population of 20 than with a population of 100, excluding MCNS and NS with recombination in large maps due to few feasible runs.

5.3 Impact of Initial Seeds

In highly constrained problems, such as the larger maps in the presented experiments, both unconstrained novelty search and MCNS are shown to perform poorly: unconstrained novelty search explores predominantly infeasible space, finding feasible individuals with difficulty but discarding them with ease, while MCNS performs random search which rarely discovers feasible individuals. For such cases, Lehman and Stanley (2010) suggest that the initial population should be "seeded" with feasible individuals to jump-start evolution. In order to test the performance of constrained and unconstrained novelty search with seeded initial populations, two experimental setups are devised: the $seed_{one}$ experiment, where a single feasible individual is injected into the initial population, and the $seed_{all}$ experiment, where every initial population consists of randomly generated individuals guaranteed to be feasible. For $seed_{one}$ experiments, the same hand-crafted individual (see Fig. 6) is injected into an otherwise randomly generated population for all runs and all methods of that map size. For $seed_{all}$ experiments, the random initial individuals are guaranteed to be feasible as they do not contain any impassable tiles that could block paths between bases and resources. These random individuals are generated by applying the repair algorithm on a blank map containing only passable tiles; resulting maps, as shown in Fig. 6, possess the mini-

		$seed_{all}$ init. population		$seed_{one}$ init. population	
Map Size	Method	p	$\bar{d}_{vis}$	p	$\bar{d}_{vis}$
Small	NS	**30.4 (12.7)**	0.506 (0.012)	**16.2 (7.5)**	0.521 (0.029)
	MCNS	**97.5 (1.7)**	**0.403 (0.019)**	**97.1 (2.2)**	**0.403 (0.018)**
	FINS	**58.7 (12.1)**	**0.448 (0.032)**	**52.8 (5.9)**	**0.466 (0.014)**
	FI2NS	**49.1 (5.4)**	0.509 (0.012)	**49.2 (4.3)**	0.519 (0.012)
Medium	NS	**35.2 (11.7)**	**0.460 (0.010)**	**1.8 (2.3)**	0.291 (0.228)
	MCNS	**97.4 (2.0)**	**0.375 (0.013)**	**96.3 (2.0)**	0.343 (0.018)
	FINS	66.7 (14.5)	**0.404 (0.045)**	**50.6 (4.5)**	**0.417 (0.013)**
	FI2NS	63.1 (16.1)	**0.434 (0.063)**	**47.4 (3.9)**	**0.481 (0.013)**
Large	NS	**24.6 (9.5)**	0.399 (0.014)	**0.0 (0.1)**	0.000 (0.000)
	MCNS	**97.0 (1.9)**	**0.315 (0.014)**	**95.5 (2.5)**	**0.274 (0.020)**
	FINS	53.4 (8.6)	**0.343 (0.015)**	**49.4 (3.7)**	**0.360 (0.020)**
	FI2NS	52.3 (11.1)	0.391 (0.044)	**45.4 (3.6)**	**0.431 (0.011)**

Table 4: Performance metrics with seeded initial populations for different novelty search approaches used with 2-point crossover and 1% chance of mutation. Since the initial population is seeded with at least one feasible individual, $n = 50$ and $g = 0$ for all methods and are omitted. Results are averaged across 50 runs, with standard deviation in parentheses. Significantly different values in the performance metrics compared to all other methods for the same map size are displayed in bold, while (significantly) best (highest) values for each metric are underlined.

mum amount of allowed bases and resources. Tables 4 and 5 present the final number of feasible individuals and their diversity for experiments with initial populations of the two types of seeds.

In experiments using 2-point crossover and 1% chance of mutation, displayed in Table 4, the final number of feasible individuals is significantly larger for $seed_{all}$ initial populations than for $seed_{one}$ or unseeded initial populations from Table 1 for 9 out of 12 cases. This is mostly due to the fact that the $seed_{all}$ initial population contains no impassable tiles; as only mutation can add impassable tiles and the chance of mutation is low, even after 100 generations maps are unlikely to have numerous impassable tiles. This benefits NS the most, as it retains a sizable feasible population even for large maps when starting from a $seed_{all}$ initial population; this is not true for $seed_{one}$ or unseeded initial populations, where NS has very few, if any, feasible individuals for medium and large maps. Since recombination applied to a $seed_{all}$ initial population results in fewer impassable tiles in the final evolved maps, the final diversity of maps for $seed_{all}$ initial populations is usually lower than that of $seed_{one}$ initial populations or even unseeded ones. For all map sizes, however, FI2NS and NS have significantly higher diversities than FINS and MCNS for $seed_{all}$ initial populations. For $seed_{one}$ initial populations, there are very few differences in the final number of feasible individuals compared to the unseeded initial populations of Table 1. However, since a feasible individual jump-starts evolution in highly constrained spaces where MCNS and FI2NS normally struggle to discover feasible individuals, these methods have more generations to perform novelty search in feasible space. This results in a significantly higher diversity for MCNS with a $seed_{one}$ initial population than with an unseeded one for medium and large maps, while FI2NS has a significantly higher diversity than MCNS and FINS even in large maps, where it underperformed with unseeded initial populations.

In experiments using mutation of a single parent, displayed in Table 5, the behavior of the different methods is overall not very different from those of Table 2. Uncon-

		$seed_{all}$ init. population		$seed_{one}$ init. population	
Map Size	Method	p	$\bar{d}_{vis}$	p	$\bar{d}_{vis}$
Small	NS	**11.5 (4.8)**	0.555 (0.019)	**11.0 (4.3)**	0.602 (0.025)
	MCNS	**84.5 (5.1)**	**0.548 (0.006)**	**79.3 (6.1)**	**0.574 (0.019)**
	FINS	40.8 (4.0)	0.554 (0.007)	38.6 (4.1)	0.604 (0.006)
	FI2NS	41.9 (4.7)	0.558 (0.006)	38.8 (4.3)	0.603 (0.010)
Medium	NS	**1.6 (1.5)**	**0.324 (0.266)**	**1.6 (1.4)**	**0.358 (0.287)**
	MCNS	**60.6 (6.9)**	0.534 (0.006)	**50.5 (6.6)**	**0.555 (0.009)**
	FINS	31.6 (5.2)	0.539 (0.009)	26.9 (5.3)	**0.573 (0.013)**
	FI2NS	30.5 (5.4)	0.535 (0.010)	25.5 (4.5)	**0.564 (0.015)**
Large	NS	**1.3 (1.1)**	**0.272 (0.243)**	**0.2 (0.6)**	0.199 (0.313)
	MCNS	**71.6 (7.9)**	0.500 (0.005)	**26.6 (5.9)**	0.490 (0.019)
	FINS	34.4 (6.1)	0.498 (0.008)	14.4 (4.2)	0.472 (0.056)
	FI2NS	33.0 (6.8)	0.496 (0.008)	13.1 (3.5)	0.467 (0.052)

Table 5: Performance metrics with seeded initial populations for different novelty search approaches used with mutation only. Since the initial population is seeded with at least one feasible individual, $n = 50$ and $g = 0$ for all methods and are omitted. Results are averaged across 50 runs, with standard deviation in parentheses. Significantly different values in the performance metrics compared to all other methods for the same map size are displayed in bold, while (significantly) best (highest) values for each metric are underlined.

strained novelty search does not manage to maintain an adequate number of feasible individuals, even with a $seed_{all}$ initial population; this is likely due to the larger number of impassable tiles introduced via mutation which increases the likelihood of blocked paths between bases and resources. The feasible diversity scores are quite similar between methods applied on the same map size, excluding NS for medium and large maps. It is not surprising that MCNS, FINS and FI2NS have a significantly larger number of final feasible individuals for experiments with $seed_{all}$ initial populations than with unseeded populations. It is surprising however that for medium and large maps, the number of feasible individuals with $seed_{one}$ initial populations is significantly lower than that with unseeded ones in 3 out of 8 cases; this is likely due to the larger number of mutations applied to the already "crowded" hand-crafted maps, which more easily create disconnected paths.

5.4 Key Findings

The argument for two-population novelty search methods builds on four hypotheses:

According to *H1*, FINS can find feasible individuals faster than FI2NS, NS and MCNS in highly constrained search spaces. Sections 5.1 and 5.2 test that hypothesis in medium and large maps, which are unlikely to be feasible by chance. *H1* is validated in 14 out of 20 experiments for the highly constrained spaces of medium and large maps, as FINS discovers feasible individuals at significantly lower g values than other methods in these 14 cases, regardless of genetic operators used; significance was not achieved in the remaining cases primarily due the high standard deviation in g of other approaches for large maps using recombination. Granted that in 4 of the 6 cases where significant differences in g were not found, the difference in n between FINS and the other methods was overwhelming, *H1* can arguably be considered validated.

According to *H2*, FINS and FI2NS can find more feasible individuals than NS. With FINS and FI2NS using the offspring boost, *H2* was validated in all 42 cases examined,

as NS had significantly fewer feasible individuals than all other methods for all experiments of Sections 5.1, 5.2 and 5.3. On the other hand, MCNS had significantly more feasible individuals than other methods in all but one of these 42 cases. Therefore, while *H2* is validated, FINS and FI2NS have fewer feasible solutions than MCNS.

According to *H3*, MCNS has lower diversity than all other methods. Since in most cases NS did not have any feasible individuals for map sizes other than small, we will test whether MCNS has a lower diversity than FINS or FI2NS. Testing experiments of Sections 5.1, 5.2 and 5.3, *H3* was validated in 28 out of 42 cases regarding FINS having a significantly higher diversity than MCNS, and again in 28 out of 42 cases regarding FI2NS having a significantly higher diversity than MCNS. The experiments which did not validate *H3* were mostly with small populations of 20 and 50 individuals, where *H3* was validated only in 1 and 3 of 12 cases for FINS and FI2NS respectively. While not validated in its entirety, *H3* seems to hold in larger populations regardless of genetic operators or initial populations used.

According to *H4*, FINS and FI2NS are more efficient in terms of population size and diversity when the offspring boost mechanism is applied. Experiments in Section 5.1 validated *H4* in all 12 cases regarding boost significantly increasing the feasible population's size and in 10 out of 12 cases regarding boost significantly increasing the diversity of feasible solutions. These results largely validate *H4*.

As a more general conclusion from the experiments of Section 5, it is clear that choosing a novelty search method depends as much on the shape of the feasible space as on the intended outcomes of the experiment. MCNS is able to create many feasible individuals which are often not particularly diverse, since it discards infeasible offspring of feasible parents. In highly constrained spaces MCNS performs poorly as it cannot find a feasible individual using an unguided, random search. On the other hand, MCNS can explore large feasible spaces efficiently, requiring a smaller population than other methods. Unconstrained NS can create diverse feasible individuals but it has no notion of the value of feasible solutions, and thus it is likely to discard them in favor of infeasible ones. This leads to few feasible solutions even when constraints are easily satisfied, but is particularly detrimental in highly constrained spaces as NS rarely retains even a single feasible solution. FI2NS shares many of the properties of NS, including the way it searches the infeasible space; as it has a notion of the value of feasible individuals, however, it can retain them and increase their numbers due to the offspring boost mechanism. Without the offspring boost, FI2NS does not fare much better than NS in highly constrained spaces; with the offspring boost, however, FI2NS usually has a sizable population of feasible individuals with a diversity rivaling that of NS. FINS retains fewer feasible individuals than MCNS and their diversity is often lower than that of FI2NS or NS; however, FINS can discover feasible individuals easily in highly constrained spaces as its infeasible population searches towards the border of feasibility. This advantage of FINS over other methods can be somewhat mitigated by injecting feasible individuals in the initial population; in such cases, FI2NS and, under conditions, MCNS can be equally or better suited at finding diverse feasible solutions.

The choice of genetic operators similarly depends on the intended outcomes of the experiment, but also on the genotype to phenotype mapping and on the distance metric used. Experiments in this paper have demonstrated that evolving solely via mutation leads to more diverse feasible individuals and a faster, more consistent discovery of feasible solutions in highly constrained spaces. However, as will be discussed in Section 6.1, the high diversity of experiments with mutation could be an artifact of the direct mapping between genotype and phenotype and of the tile-based distance metric used

to calculate diversity.

6 Discussion

The experiments presented in this paper are a first comprehensive attempt to study the behavior of novelty search in constrained search spaces. The choice of representation, distance characterization and performance metrics greatly influences the conclusions drawn; therefore, more experimentation with alternative metrics and in different domains could further validate the hypotheses put forth in Section 5. Section 6.1 discusses the limitations found within the current study and in the current experimental setup, and suggests alternative performance metrics and other distance characterizations. Section 6.2 provides some suggestions for other case studies in constrained novelty search.

6.1 Limitations & Extensions

Experiments in this paper have shown that generating offspring via mutation alone can lead to more diverse feasible individuals and a faster discovery of feasible solutions in highly constrained spaces. Mutation, by its very nature, may be more efficient at exploration than the convex search of recombination operators, but the high diversity scores it attains for most novelty search methods can easily be traced to the direct genotype to phenotype mapping and to the distance metric (d_{vis}), which directly compares phenotypical appearance. Due to these factors, novelty search essentially rewards individuals with different genotypes; thus even if mutation is destructive (e.g. rendering many feasible individuals infeasible or making substantial changes to the genotype), it would lead to higher diversity scores than a less destructive crossover operator. The current mutation operator seems to be both harmful, as it results in fewer feasible individuals than recombination, and beneficial, as it discovers feasible individuals faster and more reliably than recombination. Before any conclusions can be drawn regarding the reliability of mutation for novelty search, further experimentation should test its behavior with different mappings from genotype to phenotype and, more importantly, with different distance metrics which do not rely only on phenotypical appearance.

Experiments presented in this paper demonstrate broadly how the different methods of novelty search, both constrained and unconstrained, perform in different types of constrained spaces. However, there are several directions for enhancing this first systematic study of constrained exploration. The previous paragraph already noted that more experiments should test how mutation and recombination perform when search targets diversity in a dimension less tied to genotypical form. Depending on the distance metric used, mutation could prove less efficient or demonstrate larger deviations in the behavior of different search methods. A distance metric could also include a notion of feasibility, e.g. by assigning a high distance score when comparing feasible content with infeasible content. Such a distance metric would likely impact unconstrained novelty search, which mostly suffered in the current experiments from a distance metric lacking any indication of the value of feasible individuals. Apart from different distance metrics or genotype to phenotype mappings, other performance metrics could also be introduced for measuring the exploration of a search space. Measuring the diversity of past discoveries, such as the novel archive or populations of previous generations, could evaluate a "temporal" aspect of exploration; a high final feasible diversity, for instance, is less valuable if the entire evolutionary run was spent exploring the same area of feasible space. Including such past discoveries in a measure of exploration could warrant experiments testing the impact of the size of the novel archive

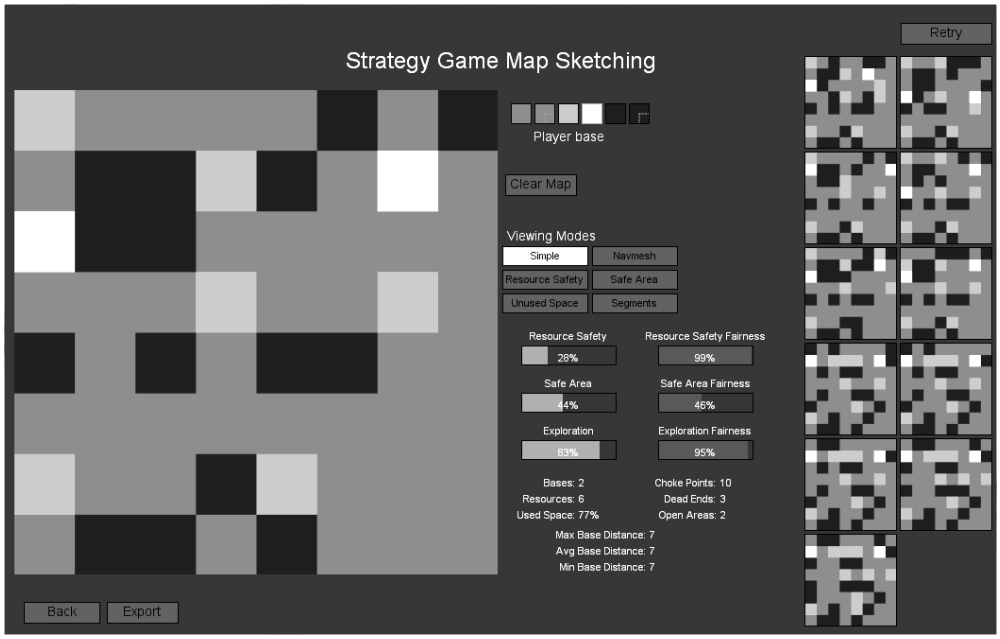


Figure 7: The user interface of Sentient Sketchbook as a human designer edits their sketch (left) and a generator, acting as the artificial designer, creates map suggestions to the user's sketch (far right). Human designers can at any time replace their current sketch with a computer-generated suggestion, and edit it further to their liking.

and the number of closest individuals used for evaluating the novelty score of Eq. 1. Preliminary tests with different numbers of novel individuals stored in the archive per generation, or with different values of k in Eq. 1 did not seem to impact the average feasible diversity in any meaningful way, and were omitted from this paper.

6.2 Other Applications of Constrained Novelty Search

While this paper applied constrained novelty search in the domain of automated generation of game content, there are a number of other domains which would benefit from constrained novelty search.

Computer-aided design tools such as Sentient Sketchbook (Liapis et al., 2013d) can use constrained novelty search to generate suggestions for their users. Using the same map sketch format as presented in this paper, Sentient Sketchbook generates and presents suggestions in real-time (see Fig. 7), while the user is sketching a game level. Presented suggestions are evolved from a small initial population consisting of mutations of the current user sketch, which ensures that most of the map structure (such as number of bases or placement of impassable tiles) of the user's sketch is maintained. This initial population is further diversified via constrained novelty search to maximize the maps' visual diversity. As constrained novelty search runs for a few generations, in order to ensure that suggestions are generated more or less in real-time, the most diverse evolved maps shown to the user will have certain similarities but also differences with their initial sketch. Such suggestions are not so different from the current user sketch to be deemed randomly generated; this local exploration of the search space is expected to enhance the user's creativity as they are presented with viable alterna-

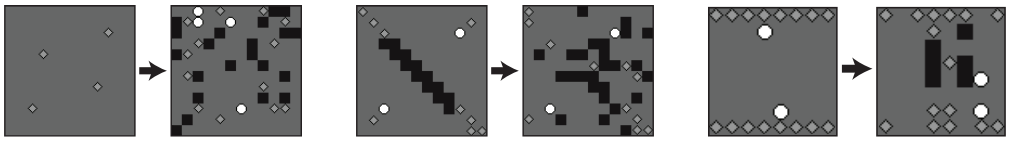


Figure 8: Map sketches authored by industry experts of Sentient Sketchbook (left) and the generated suggestions selected by these users to replace their designs (right). All shown suggestions were generated via novelty search on a population of permutations of the user's sketch. Some map integrity with the user's sketch is maintained in the suggestions (see resource placement on the edges of Fig. 8b and 8c), although it is clear that divergent search has also enhanced the visual diversity of suggestions compared to the initial user sketch. These suggestions are guaranteed to be playable, even if the user sketches they evolved from were not, as is the case with Fig. 8a.

tives to their current design. Moreover, the guarantee that presented suggestions are playable — as they satisfy all imposed constraints — makes them useful to the designer. In an earlier evaluation of the Sentient Sketchbook tool with expert users working in the game industry, novel suggestions were preferred over objective-driven ones in early steps of the design process, where users lacked inspiration for a strategy game level (see Fig. 8). Novelty search used what little information was provided by the user, such as a few resources placed randomly around the map as in Fig. 8a, to add bases and impassable tiles and create a complete, playable map. Acting as a creative "spark", these maps were then further edited by the users to conform to their own notion of game quality. The current version of Sentient Sketchbook uses FINS with recombination as described in Section 4, due to the requirement for fast discovery of feasible individuals.

Constrained novelty search does not need to rely on a human user to be creative, however. As constrained novelty search generates content which can be both *novel* and *useful* (by fulfilling some constraints on "usefulness") it can be considered a creative process in itself (Boden, 1990). In order to be truly creative, however, such a process should be able to understand what drives its search. This can be accomplished by adapting its characterization of diversity according to the artifacts it is currently creating. As this characterization drives the exploration of the search space, adapting it to break current patterns of its artifacts will lead to the exploration of the search space in a new dimension, orthogonal to the one currently being explored. This will not only result in a more thorough exploration of the search space but also simulates, to a degree, the way humans are creative via lateral thinking (De Bono, 1970) and reinterpreting creative problems (Grace et al., 2013). Constrained novelty search has been applied in preliminary work exploring the potential of adapting the characterization of diversity by learning the distinct patterns of evolved artifacts (Liapis et al., 2013b); these patterns were ascertained via autoencoders trained on a set of spaceships evolved via constrained novelty search. Ensuring that the generated spaceships are viable (i.e. of sufficient size and with no disconnected pixels), the exploration of the feasible search space was enhanced by the adaptive characterization of divergence, which drove the search towards discovering patterns previously unseen in the population.

A domain where novelty search has been used quite successfully in recent years is robot locomotion; the work of Risi and Stanley (2013) and Morse et al. (2013) uses novelty search to evolve quadruped robot controllers via HyperNEAT (Stanley et al.,

2009). Unlike the search for visual diversity used in this study, these robot controllers evolve to diversify a *behavioral* characterization, e.g. the robots' positions sampled at certain intervals during a simulation. There are a number of constraints on the behavior of a robot controller: for instance, Risi and Stanley (2013) stop the simulation which evaluates a robot controller if the robot's torso falls under a specific height. Currently controllers deemed infeasible are assigned a novelty score of zero, amounting to minimal criteria novelty search. However, such infeasible controllers can be retained in a separate population which evolves either to maximize the diversity of their behavior via FI2NS or to minimize their distance from feasibility via FINS; the latter can be accomplished by awarding a high fitness to individuals which "fall" at a later time step than others, or which can return to an upright position after falling.

7 Conclusions

This paper elaborated on the topic of constrained novelty search and proposed two-population novelty search methods for exploring both feasible and infeasible spaces. These approaches evolve two populations simultaneously: a feasible population to maximize novelty and an infeasible population either to minimize the distance from feasibility (FINS) or to maximize novelty (FI2NS). These two-population methods of constrained novelty search were built on a number of hypotheses; FINS was assumed to discover feasible individuals faster as it searches the infeasible space efficiently, while FI2NS was assumed to be able to discover disjoint sections of feasible space and thus create more diverse feasible solutions. These hypotheses were tested in the domain of game content generation, with the goal of discovering and diversifying playable game levels. Comparing FINS and FI2NS with existing methods of constrained and unconstrained novelty search, our hypotheses were validated for the majority of experimental setups; the few cases where the hypotheses were not validated were attributed to small population sizes, easily satisfied constraints and erratic behavior of the other novelty search methods being tested. Finally, boosting the number of offspring in the feasible population is shown to be beneficial in increasing both the number and the diversity of feasible individuals in FINS and especially in FI2NS.

Both FINS and FI2NS seem particularly useful for the procedural generation of game content, which requires its generated artifacts to be diverse yet playable. Diversity is important for game elements, such as levels, because the personal tastes of different players are difficult to capture via mathematically defined objectives. Depending on the computational requirements of the content generators, such as *offline* while the game loads or *online* while the game is played, the higher diversity of FI2NS or the speedy discovery of feasible content of FINS may be preferable. Outside of game content generators, constrained novelty search can also be directly used as a model of computational creativity since it creates both novel and useful artifacts, as a tool for exploring different behaviors of robot controllers, and in any other domain where constraint satisfaction is essential.

8 Acknowledgments

This paper adds significant value to the work of Liapis et al. (2013c), which introduced two-population novelty search methods and tested them on the domain of game level generation. Research was supported, in part, by the FP7 ICT project C2Learn (project no: 318480) and by the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

Boden, M. (1990). *The Creative Mind*. Abacus.

Browne, C. and Maire, F. (2010). Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16.

Cardamone, L., Yannakakis, G. N., Togelius, J., and Lanzi, P. L. (2011). Evolving interesting maps for a first person shooter. In *EvoApplications (1)*, pages 63–72.

Coello Coello, C. A. (2010). Constraint-handling techniques used with evolutionary algorithms. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 2603–2624. ACM.

De Bono, E. (1970). *Lateral thinking: creativity step by step*. Harper & Row.

Dunn, O. (2012). Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64.

Grace, K., Gero, J., and Saunders, R. (2013). Learning how to reinterpret creative problems. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 113–117.

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

Kramer, O. and Schwefel, H.-P. (2006). On three new approaches to handle constraints within evolution strategies. *Natural Computing*, 5(4):363–385.

Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the International Conference on Artificial Life (ALIFE XI)*, pages 329–336. MIT Press.

Lehman, J. and Stanley, K. O. (2010). Revising the evolutionary computation abstraction: Minimal criteria novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 103–110.

Lehman, J. and Stanley, K. O. (2011a). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.

Lehman, J. and Stanley, K. O. (2011b). Improving evolvability through novelty search and self-adaptation. In *IEEE Congress on Evolutionary Computation*, pages 2693–2700.

Lewis, M. (2008). Evolutionary visual art and design. In *The Art of Artificial Evolution*, pages 3–37.

Liapis, A., Martínez, H. P., Togelius, J., and Yannakakis, G. N. (2013a). Adaptive game level creation through rank-based interactive evolution. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*.

Liapis, A., Martínez, H. P., Togelius, J., and Yannakakis, G. N. (2013b). Transforming exploratory creativity with DeLeNoX. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 56–63.

Liapis, A., Yannakakis, G., and Togelius, J. (2013c). Enhancements to constrained novelty search: Two-population novelty search for generating game content. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 343–350.

Liapis, A., Yannakakis, G., and Togelius, J. (2013d). Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*, pages 213–220.

Liapis, A., Yannakakis, G. N., and Togelius, J. (2011). Neuroevolutionary constrained optimization for content creation. In *Proceedings of IEEE Conference on Computational Intelligence and Games*, pages 71–78.

Liapis, A., Yannakakis, G. N., and Togelius, J. (2012). Adapting models of visual aesthetics for personalized content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):213–228.

Liapis, A., Yannakakis, G. N., and Togelius, J. (2013e). Generating map sketches for strategy games. In *Proceedings of Applications of Evolutionary Computation*, volume 7835, LNCS, pages 264–273. Springer.

Liapis, A., Yannakakis, G. N., and Togelius, J. (2013f). Sentient world: Human-based procedural cartography. In *Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt)*, volume 7834, LNCS, pages 180–191. Springer.

Liapis, A., Yannakakis, G. N., and Togelius, J. (2013g). Towards a generic method of evaluating game levels. In *Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*.

Michalewicz, Z. (1995a). Do not kill unfeasible individuals. In *Proceedings of the Fourth Intelligent Information Systems Workshop*, pages 110–123.

Michalewicz, Z. (1995b). A survey of constraint handling techniques in evolutionary computation methods. In *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 135–155.

Michalewicz, Z., Dasgupta, D., Le Riche, R., and Schoenauer, M. (1996). Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30:851–870.

Moraglio, A. (2011). Abstract convex evolutionary search. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*, pages 151–162. ACM.

Morse, G., Risi, S., Snyder, C. R., and Stanley, K. O. (2013). Single-unit pattern generators for quadruped locomotion. In *Proceeding of the Genetic and Evolutionary Computation Conference*, pages 719–726. ACM.

Risi, S., Lehman, J., D'Ambrosio, D. B., Hall, R., and Stanley, K. O. (2012). Combining search-based procedural content generation and social gaming in the petalz video game. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012)*.

Risi, S. and Stanley, K. O. (2013). Confronting the challenge of learning a flexible neural controller for a diversity of morphologies. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 255–262. ACM.

Schoenauer, M. and Michalewicz, Z. (1996). Evolutionary computation at the edge of feasibility. In *Proceedings of the 4th Parallel Problem Solving from Nature*, pages 245–254.

Shaker, N., Yannakakis, G. N., and Togelius, J. (2010). Towards automatic personalized content generation for platform games. In *Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*, pages 63–68.

Sorenson, N., Pasquier, P., and DiPaola, S. (2011). A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):229–244.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artifiicial Life*, 15(2):185–212.

Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelbäck, J., Yannakakis, G. N., and Grappiolo, C. (2013). Controllable procedural map generation via multiobjective evolution. *Genetic Programming and Evolvable Machines*, 14(2):245–277.

Togelius, J., Yannakakis, G., Stanley, K., and Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.

Yannakakis, G. N. (2012). Game AI revisited. In *Proceedings of ACM Computing Frontiers Conference*, pages 285–292.

Yannakakis, G. N., Liapis, A., and Alexopoulos, C. (2014). Mixed-initiative co-creativity. In *Proceedings of the ACM Conference on Foundations of Digital Games*.

Yannakakis, G. N. and Togelius, J. (2011). Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 2(3):147–161.