Theoretical Analysis of Evolutionary Algorithms with Quality Diversity for a Classical Path Planning Problem

Duc-Cuong Dang[1], **Aneta Neumann**[2], **Frank Neumann**[2], **Andre Opris**[1], **Dirk Sudholt**[1]
[1]Chair of Algorithms for Intelligent Systems, University of Passau, Germany
[2]Optimisation and Logistics, The University of Adelaide, Australia

Abstract

Quality diversity (QD) algorithms, an extension of evolutionary algorithms, excel at generating diverse sets of high-quality solutions for complex problems in robotics, games, and combinatorial optimisation. Despite their success, the underlying mechanisms remain poorly understood due to a lack of a theoretical foundation. We address this gap by analysing QD algorithms on the all-pairs-shortest-paths (APSP) problem, a classical planning task that naturally seeks multiple solutions. Using Map-Elites, a prominent QD approach, we leverage its ability to evolve solutions across distinct regions of a behavioural space, which for APSP corresponds to all pairs of nodes in the graph.

Our analysis rigorously demonstrates that evolutionary algorithms using Map-Elites efficiently compute shortest paths for all node pairs in parallel by exploiting synergies in the behavioural space. By appending edges to an existing shortest path, mutation can create optimal solutions in other regions of the behavioural space. Crossover is particularly effective, as it can combine optimal paths from two regions to produce an optimal path for a third region simply by concatenating two shortest paths. Finally, refining the parent selection to facilitate successful crossovers exhibits significant speed-ups compared to standard QD approaches.

1 Introduction

In recent years, computing diverse sets of high quality solutions for combinatorial optimisation problems has gained significant attention in the area of artificial intelligence from both theoretical [Baste *et al.*, 2022; Baste *et al.*, 2019; Fomin *et al.*, 2024; Hanaka *et al.*, 2023] and experimental [Vonásek and Saska, 2018; Ingmar *et al.*, 2020] perspectives. Prominent examples where diverse sets of high quality solutions are sought come from the area of path planning [Hanaka *et al.*, 2021; Gao *et al.*, 2022].

Evolutionary algorithms provide a flexible way of generating diverse sets of high-quality solutions by directly incorporating diversity measures into the population. Generating diverse and high-quality solution sets has gained significant interest in evolutionary computation, particularly under the notion of evolutionary diversity optimisation (EDO) [Gao and Neumann, 2014; Gao *et al.*, 2021; Neumann *et al.*, 2018] and quality diversity (QD) [Lehman and Stanley, 2011; Mouret and Clune, 2015; Hagg *et al.*, 2018].

EDO algorithms maintain a fixed population size and focus on maximising diversity based on a specified diversity metric by ensuring that all solutions meet a given quality criteria. In contrast, QD algorithms typically utilise a variable population size to identify optimal solutions across different niches within a specified behavioural space. Approaches that use a multidimensional archive of phenotypic elites, called Map-Elites [Mouret and Clune, 2015], are among the most commonly used QD algorithms. Both approaches have initially been applied to design problems [Hagg *et al.*, 2018; Neumann *et al.*, 2019]. QD algorithms have shown to produce excellent results for challenging problems in the areas such as robotics [Miao *et al.*, 2022; Shen *et al.*, 2020], games [Cully and Demiris, 2018] and combinatorial optimisation [Nikfarjam *et al.*, 2024a]. In fields such as robotics and gaming, several variants of QD algorithms have been developed [Pugh *et al.*, 2016; Gravina *et al.*, 2018; Fontaine *et al.*, 2020; Zardini *et al.*, 2021; Fontaine *et al.*, 2021; Medina *et al.*, 2023]. Moreover, QD has been used to evolve instances for the traveling salesperson problem [Bossek and Neumann, 2022], for solving the traveling thief problem [Nikfarjam *et al.*, 2024a], and in context of time-use optimisation to improve health outcomes [Nikfarjam *et al.*, 2024b].

Despite these empirical success stories, the development of QD remains in its early stages. It is not well understood when and why QD is effective, nor how long it takes to evolve diverse and high-quality solutions for meaningful problems. Due to the lack of a solid theoretical foundation, fundamental questions remain unanswered, including which evolutionary operators are most effective and how to design the most efficient QD algorithms for specific problem classes.

In contrast, for traditional evolutionary algorithms that optimise a single objective, a robust theoretical foundation has been established over the past two decades through *runtime analysis*. This approach examines the random time required to evolve optimal solutions for a wide range of benchmark problems and combinatorial optimisation [Neumann and Witt, 2010; Jansen, 2013; Doerr and Neumann, 2020], as well as the benefits of diversity mechanisms [Dang *et al.*, 2016; Sudholt, 2020; Ren *et al.*, 2024]. However, the goals of QD go beyond finding a global optimum. The aim is to evolve a population of diverse solutions and to simultaneously optimise multiple regions of the behavioural space. This is related to (but not the same as) *multimodal optimisation* [Friedrich *et al.*, 2009; Covantes Osuna and Sudholt, 2022].

Rigorous theoretical studies of QD algorithms have only begun recently. The first runtime analysis was conducted for the classical knapsack problem [Nikfarjam *et al.*, 2022], demonstrating that QD can solve the problem in expected pseudo-polynomial time. Subsequent runtime guarantees have been established for the computation of minimum spanning trees and the optimisation of submodular monotone functions under cardinality constraints [Bossek and Sudholt, 2024], as well as their generalisation to approximately submodular functions and the classical set cover problems [Qian *et al.*, 2024]. These studies share a common focus: they aim to identify a single global optimum (or good approximation, respectively), while leveraging Map-Elites as a framework to store intermediate solutions that aid in finding the global optimum. The only study to analyse QD algorithms for problems that genuinely seek multiple solutions was conducted in [Schmidbauer *et al.*, 2024]. The authors considered monotone submodular functions with artificial Boolean constraints that defined subproblems, which in turn acted as stepping stones to address the main optimisation problem.

1.1 Our Contribution

We investigate and demonstrate the effectiveness of QD on a classical planning problem that is inherently well-suited to QD, as it naturally involves evolving many diverse solutions. The all-pairs shortest paths (APSP) problem seeks to find shortest paths between all pairs of nodes in a graph. From a QD perspective, the behavioural space is naturally defined by all pairs of nodes in the graph, where each pair u, v of nodes corresponds to the task of finding a shortest path between u and v. Our study examines the performance of QD algorithms on the APSP problem to gain insights into the fundamental working principles of QD. Additionally, we aim to identify the most effective evolutionary operators and explore ways to enhance the efficiency of QD algorithms.

The APSP can be solved in polynomial time using classical algorithms [Floyd, 1962; Johnson, 1977]. The goal of the investigations of evolutionary computation techniques for this problem is not to beat these classical algorithms in terms of runtime, but to provide a theoretical understanding of their working behaviour for this important fundamental problem.

Our work builds on the previous analysis in [Doerr *et al.*, 2012] for so-called ($\leq$ μ+1) evolutionary algorithms. It showed that introducing crossover as an additional operator can reduce the expected runtime on n-vertex graphs to $O(n^{3.5}\sqrt{\log n})$, compared to a runtime of at least $\Omega(n^4)$ when relying solely on mutation. The upper bound was improved to $O(n^{3.25}\log^{1/4} n)$ in [Doerr and Theile, 2009] with a more refined analysis (which was also shown to be asymptotically tight in the worst case), and later to $O(n^3 \log n)$ in [Doerr *et al.*, 2013] using better operators. Similar speedups through crossover were also obtained for ant colony optimisation algorithms [Sudholt and Thyssen, 2012]. Despite criticism of the ($\leq$ μ+1) scheme at the time, e. g. see [Corus and Lehre, 2018], we believe such a scheme is natural in the modern view of QD algorithms because not-yet existing solutions (those not-yet occupying their slots) in the population are equivalent to empty cells in the archive of a Map-Elites algorithm. We therefore define and analyse the QD-GA algorithm, in which a map of n by n is maintained to store the best-so-far shortest paths between all source-destination pairs for APSP. In each iteration, crossover or mutation operators as defined in [Doerr *et al.*, 2012] are exclusively applied from parents uniformly selected from the cells to create a new offspring solution which can be used to update the map. When only mutation is used, the algorithm is referred to as QD-EA.

Using a similar approach as in previous work, however with a slightly different tool from [Witt, 2014], we show that QD-EA using only mutation optimises APSP in $O(n^2\Delta \max\{\ell, \log n\})$ fitness evaluations in expectation. Here Δ is the maximum degree and ℓ is the diameter of the graph. We find that mutation can exploit synergies between different parts of the behavioural space since appending an edge to an existing shortest path can create shortest paths in other regions of the behavioural space. Crossover is particularly effective at exploiting synergies as it can simply concatenate two shortest paths to produce an optimal path in a third region. We prove that QD-GA with a constant probability of applying crossover guarantees an expected runtime in $O(\Delta^{3/4} n^{5/2} \log^{1/4} n)$. This refines the previous bound of $O(n^{3.25}\log^{1/4} n)$ from [Doerr and Theile, 2009] by taking into account the maximum degree Δ of the graph and yields better guarantees for graphs with small maximum degree. For instance, on graphs with $\Delta = O(1)$ our new runtime bound is smaller by a factor of order $n^{3/4}$.

Finally, we prove that if the selection of parent cells is restricted to only include *compatible* parents for crossover from the map, QD only relying on crossover optimises APSP in $O(n^3 \log n)$ expected fitness evaluations, and therefore when combining with mutation, the expected runtime is $O(\min\{n^2\Delta \max\{\ell, \log n\}, n^3 \log n\})$. We refer to this latter version of QD-GA as the Fast QD-APSP algorithm. Our runtime bounds hold with high probability and in expectation. We also provide a worst-case instance and a lower bound showing that the upper bound of $O(n^3 \log n)$ expected fitness evaluations is best possible.

2 Quality Diversity and the APSP

The all-pairs-shortest-path (APSP) problem is a classical combinatorial optimisation problem. Given a directed strongly connected graph $G(V, E)$ with $n = |V|$ and a weight function $w\colon E \to \mathbb{N}$ on the edges, the goal is to compute for any given pair of nodes $(s, t) \in V \times (V \setminus \{s\})$, a shortest path (in terms of the weight of the chosen edges) from s to t. If one only looks for the shortest path between two specific nodes, the problem is referred as the single-source single-destination shortest-path problem (SSSDSP). Like in [Doerr *et al.*, 2012], we assume that G is strongly connected, thus there exists a path from s to t for any distinct pair (s, t) of nodes.

From a QD perspective, we see solving APSP as evolving

Algorithm 1: Quality Diversity Genetic Algorithm (QD-GA) with Crossover Probability p_c

```
Input: graph $G = (V, E)$
1  Initialise an empty map $M$;
2  Set $M_{s,t} = (s, t), \forall (s, t) \in E$;
3  while termination condition not met do
4      Choose $p \in [0, 1]$ uniformly at random;
5      if $p \leq p_c$ then
6          Choose $I_{st}, I_{uv} \in M$ uniformly at random;
7          Generate $I'$ from $I_{st}$ and $I_{uv}$ by crossover;
8      else
9          Choose an individual $I_{st} \in M$ uniformly at random;
10         Generate $I'$ from $I_{st}$ by mutation;
11     $M \leftarrow$ Update$(M, I')$;
```

Algorithm 2: Update Procedure for Minimization

```
1  if $I'$ is a valid path from $s$ to $t$ then
2      if $M_{st}$ is empty then store $I'$ in cell $M_{st}$;
3      else
4          Let $I$ be the search point in cell $M_{st}$;
5          if $f_{st}(I') \leq f_{st}(I)$ then replace $I$ by $I'$ in $M_{st}$;
```

both diverse and high quality solutions of multiple SSSDSP. Therefore, a valid search point I is a set of chosen edges, denoted by $E(I)$, that forms a valid path from a starting node s to a target node t. This path can also be represented by a sequence of nodes to visit, i. e. $I := (s = v_0, v_1, \dots, v_k = t)$ such that $E(I) := \{(v_{i-1}, v_i) \mid i \in [k]\} \subseteq E$ where $[n]$ denotes the set $\{1, \dots, n\}$ for $n \in \mathbb{N}$. By $|I|$ we denote its *cardinality*, that is, $|I| := |E(I)| = k$. Occasionally, to make the source s and destination t clear, we also write such path I as I_{st}. For storage, either the sequence I or the set $E(I)$ can be stored in memory as either one of these uniquely defines I. For a given search point I, let

$$f_{st}(I) = \sum_{e \in E(I)} w(e)$$

be the weight (or length) of I. A shortest path between s and t is the one that minimises f_{st}, and if this path is the above path I then $I_{v_i v_{i+j}} = (v_i, v_{i+1}, \dots, v_{i+j})$ for any $i \in \{0, \dots, k-1\}$ and any $j < k - i$ is also a shortest path between v_i and v_{i+j}.

We use the Quality Diversity Genetic Algorithm (QD-GA) given in Algorithm 1 for our investigations. The algorithm works with a 2D map M of dimension n by n excluding the diagonal. The rows and columns of M are indexed by nodes of V, and each cell $(s, t) \in V \times (V \setminus \{s\})$ in the map can store an individual I_{st} representing a path that starts at s and ends in t. Following [Doerr *et al.*, 2012], our map M is initialised to contain for each cell M_{st} where $(s, t) \in E$. the solution I with $E(I) = \{(s, t)\}$ consisting of the path from s to t given by the edge (s, t), while the other cells of M are initially empty. In each iteration, either mutation or crossover is used to produce a new individual from the individuals of the current map M. Crossover is carried out in each iteration with probability p_c and chooses two individuals from M uniformly at random to produce an offspring I'. If crossover is not applied, mutation is performed on an individual chosen uniformly at random to produce I'. I' is then used to update the map M using Algorithm 2. If I' is a valid path from some node s to some node t, then I' is introduced into the cell M_{st} if M_{st} is currently empty. Otherwise, if M_{st} is not empty, then I' replaces I_{st} in M_{st} iff $f(I') \leq f(I_{st})$ holds.

As common in the theoretical analysis of evolutionary algorithms, we measure the runtime of Algorithm 1 by the number of fitness evaluations. The *optimisation time* of Algorithm 1 refers to the number of fitness evaluations, until M contains for each pair $(s, t) \in V \times (V \setminus \{s\})$ a shortest path from s to t. The *expected optimisation time* refers to the expectation of this value. This optimisation time can depend on the following characteristics of the graph: the maximum degree $\Delta := \max_{u \in V} |\{(v \mid (u, v) \in E \vee (v, u) \in E\}|$, and the largest cardinality of the shortest paths $\ell := \max_{(s,t) \in V \times (V \setminus \{s\}), I \in \mathcal{I}(s,t)} \{|I| \mid f_{st}(I) \text{ is minimised}\}$, where $\mathcal{I}(s, t)$ denotes set of all valid paths from s to t. For unweighted graphs, this parameter ℓ is known as the *diameter*.

2.1 Mutation and Crossover

We use the following mutation and crossover operators introduced in [Doerr *et al.*, 2012]. For a given path I_{st}, starting at node s and ending at node t, let E_{st} be the set of all edges incident to s or t. Mutation relies on the elementary operation of choosing an edge $e \in E_{st}$ uniformly at random dependent on a given individual I'_{st}. If e is part of I', then e is removed from I'. Otherwise, e is added I' with potentially extends the path at its start or end node. The offspring I' is obtained by creating a copy of I_{st} and applying this elementary operation $k + 1$ times to I' where k is chosen according to a Poisson distribution $Pois(\lambda)$ with parameter $\lambda = 1$. Note that an elementary operation might create an invalid individual when adding an outgoing edge to s or an incoming edge to t. Such invalid individuals are rejected by the algorithm according to the update procedure given in Algorithm 2.

Our analysis will rely on mutation steps carrying out a single valid elementary operation extending a given path. For crossover, we choose I_{st} and I_{uv} uniformly at random from the current map M. If $t = u$, the resulting offspring I is obtained by appending I_{uv} to I_{st} and constitutes a path from s to v. Otherwise, the offspring is invalid and has no effect during the update procedure of Algorithm 2.

2.2 Analytical Tools

We use the following tail bound from [Witt, 2014] for sum of geometric variables in our analyses.

Lemma 1 (Theorem 1 in [Witt, 2014]). *Let $\{X_i\}_{i \in [n]}$ be independent random geometric variables with parameter $p_i \geq 0$, and let $X := \sum_{i=1}^{n} X_i$, and $p := \min_{i \in [n]}\{p_i\}$. If $\sum_{i=1}^{n} p_i^{-2} \leq s \leq \infty$ then for any $\lambda \geq 0$ it holds that*

$$\Pr(X \geq \mathrm{E}[X] + \lambda) \leq e^{-\frac{1}{4}\min\left(\frac{\lambda^2}{s}, \lambda p\right)}.$$

We then have the following corollary for a single variable.

Corollary 2. *Let X be a geometric random variable with parameter $p > 0$, then for any real number $c > 0$ and any natural number $n \geq e^{1/c}$, it holds* $\Pr\left(X \geq \frac{c\ln n+1}{p}\right) \leq n^{-c/4}$.

Proof. Recall $\mathrm{E}[X] = 1/p$, thus applying Lemma 1 with $\lambda := \frac{c\ln n}{p}$ and $s := \frac{1}{p^2}$ gives

$$\Pr\left(X \geq \frac{c\ln n+1}{p}\right) \leq e^{-\frac{\min\left((c\ln n)^2, c\ln n\right)}{4}} = n^{-\frac{c}{4}}. \square$$

We will also need the following inequality.

Lemma 3. *For any real number $x > 0$, it holds that* $2\lfloor x\rfloor - \lfloor(3/2)x\rfloor + 1 \geq \lceil x/2\rceil - 1$.

Proof. Applying the well-known inequality $\lfloor x+y\rfloor \leq \lfloor x\rfloor + \lfloor y\rfloor + 1$ which holds for any real numbers x and y, to $\lfloor(3/2)x\rfloor = \lfloor x + x - x/2\rfloor$ twice gives $\lfloor(3/2)x\rfloor \leq \lfloor x\rfloor + \lfloor x\rfloor + \lfloor -x/2\rfloor + 2 = 2\lfloor x\rfloor - \lceil x/2\rceil + 2$. Putting this back to the original statement concludes the proof. □

3 Exploiting Synergies via Mutation

We first consider the optimisation progress achievable through mutation only. Particularly, we refer to QD-GA (Algorithm 1) with $p_c = 0$ as QD-EA. The following theorem shows that QD-GA with a constant probability of carrying out mutation, thus including QD-EA as a special case, solves APSP efficiently and bounds its optimisation time depending on the structural parameters Δ and ℓ of the given input. Note that the algorithm does not need to know Δ nor ℓ.

Theorem 4. *The optimisation time of QD-GA with $p_c = 1 - \Omega(1)$, which includes QD-EA, is $O(n^2\Delta\max\{\ell, \log n\})$ in expectation and with probability $1 - o(1)$.*

The same holds for the time to find shortest paths between all pairs of nodes whose shortest paths have cardinality at most k, when replacing ℓ with k.

Proof. We first estimate the number of iterations T_{st} to optimise an arbitrary cell M_{st}. Let $I = (s = v_0, \ldots, v_k = t)$ be any shortest path of this cell, then let X_i be the number of iterations in which cell M_{sv_i} is optimised but $M_{sv_{i+1}}$ is not yet optimised, for any $i \in \{0, \ldots, k-1\}$, so $T_{st} = \sum_{i=0}^{k-1} X_i$. Since multiple elementary operations are allowed in one mutation, it is possible to optimise longer paths before the shorter ones, in other words some X_i can take value zero. However, for an upper bound on the optimisation time it suffices to consider the case where the path is extended by adding only one correct edge at a time. That is, to optimise $M_{sv_{i+1}}$, it suffices to pick the solution in the optimised cell M_{sv_i} as parent, i. e. probability $1/(n(n-1))$, note that this solution has an equal weight to that of the path $(v_1, \ldots, v_i)$. Then the mutation is applied with only one elementary operation, i. e. with probability $(1-p_c)/e$ for the distribution $\mathrm{Pois}(1)$, where the edge (v_i, v_{i+1}) is chosen for adding to the parent, i. e. with probability at least $1/(2\Delta)$. The obtained offspring therefore has equal weight to that of the shortest path $(v_0, \ldots, v_{i+1})$ thus it is used to update and hence optimise $M_{sv_{i+1}}$, and so variable X_i is stochastically dominated by a geometric random variable Y_i with parameter $p := (1-p_c)/(2en^2\Delta)$ regardless of i and of the target path I. Furthermore,

$$|I| = k \leq \ell \leq \max\{\ell, 5\log n\} =: \ell'$$

and therefore

$$T_{st} = \sum_{i=0}^{k-1} X_i \preceq \sum_{i=0}^{\ell'-1} Y_i =: Y.$$

By linearity of expectation $\mathrm{E}[Y] = \frac{2en^2\Delta}{1-p_c} \cdot \ell'$, thus applying Lemma 1 with $\lambda := \frac{2(6-e)n^2\Delta\ell'}{1-p_c}$ gives

$$\Pr\left(T_{st} \geq \frac{12n^2\Delta\ell'}{1-p_c}\right) \leq \Pr(Y \geq \mathrm{E}[Y] + \lambda)$$
$$\leq e^{-\frac{1}{4}\min\left(\frac{\lambda^2}{(1/p)^2\ell'}, \lambda p\right)}.$$

Note that $\lambda p = (6/e - 1)\ell'$ while $\lambda^2/((1/p)^2\ell') = (6/e-1)^2\ell' > \lambda p$, thus the probability that M_{st} is not optimised after $\frac{12n^2\Delta\ell'}{1-p_c} =: \tau$ iterations is at most $e^{-(6/e-1)\ell'/4}$.

We now consider all the $n(n-1)$ cells. Given $\ell' \geq 5\log n = 5\ln n/\ln 2$, by a union bound the probability that not all cells are optimised after τ iterations is at most

$$n(n-1)e^{-(6/e-1)\ell'/4} \leq n^2 e^{-5(6/e-1)\ln n/(4\ln 2)} < n^{-1/6}$$

since $(30/e - 5)/(4\ln 2) > 1/6$. Thus with probability $1 - o(1)$, all cells are optimised in $\tau = O(n^2\Delta\ell')$ iterations since $1 - p_c = \Omega(1)$. Particularly this holds regardless of the initial population, thus the expected optimisation time is at most $(1 + o(1))O(n^2\Delta\ell')$.

The final statement is obvious from the proof. □

Note that the running time of QD-EA heavily depends on the characteristics of the graph. Particularly, for $\Delta = \Theta(n)$ and $\ell = \Theta(n)$ we obtain an upper bound of $O(n^4)$ on the runtime of the algorithm. A matching lower bound of $\Omega(n^4)$ can be obtained for $\Delta = \Theta(n)$ and $\ell = \Theta(n)$ by considering the complete directed graph K_n as follows. It is defined as $K_n = (V, E)$ with $V := \{v_1, \ldots, v_n\}$, $E := \{(u, v) \mid u, v \in V, u \neq v\}$, and weights $w(v_i, v_j) = 1$ if $j - i = 1$ and $w(v_i, v_j) = n$ otherwise (see [Doerr *et al.*, 2012]). Using Theorem 11 in [Doerr *et al.*, 2012], we can obtain the following lower bound.

Theorem 5. *The optimisation time of QD-EA on K_n with the above weight function is $\Omega(n^4)$ with probability $1 - o(1)$ and therefore, the expected optimisation time is $\Theta(n^4)$.*

4 The Effectiveness of Crossover

We now consider the case where p_c is a constant in $(0, 1)$. This implies that both crossover and mutation are enabled. We show that crossover can significantly speed up the construction of shortest paths by concatenating shortest paths.

Following the analysis in [Doerr and Theile, 2009], we argue that when seeking a shortest path between two vertices u and v, crossover is efficient in creating optimal sub-paths that may leave small gaps near u or v, which are then efficiently filled by mutation.

Definition 6. *For $g \in \mathbb{N}_0$ a pair of vertices u, v is called g-near-complete with respect to a search point x if there exists a shortest path $u = v_0, v_1, \dots, v_k = v$ and integers $a, b \le g$ such that x contains a shortest path for the pair v_a, v_{k-b}.*

The following theorem gives an improved upper bound compared to Theorem 4 for all graphs with $\Delta^{1/4}\ell = \omega(n^{1/2}\log^{1/4} n)$. If refines the bound $O(n^{3.25}\log^{1/4} n)$ from [Doerr and Theile, 2009] as it takes into account the maximum degree Δ of the graph. For graphs with $\Delta = O(1)$ this is an improvement by a factor of order $n^{3/4}$.

Theorem 7. *The optimisation time of QD-GA with constant $p_c \in (0, 1)$ is $O(\Delta^{3/4}n^{5/2}\log^{1/4} n)$ in expectation and with probability* $1 - o(1)$.

Proof. When a cell is optimised, it can admit multiple shortest paths of various cardinalities but with the same optimised weight. After that Algorithm 2 can still update the cell with these equivalently optimal solutions. To argue about the optimisation time, we therefore consider for each cell one of its shortest paths with the largest cardinality as the *representative* optimal solution (representative for short) of the cell. Since shortest paths can never be lost, we will exploit the following argument. Throughout the proof, we focus separately on iterations that either do not execute crossover or do, while disregarding potential progress made in the other type of iteration. We account for the expected waiting times for the desired type of iteration. Since p_c is a constant in $(0, 1)$, the expected waiting times $1/p_c$ and $1/(1 - p_c)$ are constant factors that can be absorbed in the asymptotic notation.

Let $m := \lfloor \Delta^{-1/4} n^{1/2} \log^{1/4}(n) \rfloor$. We divide the run into $\lfloor \log_{3/2}(\ell/m) \rfloor + 1 \le \lfloor \log_{3/2}(n/m) \rfloor + 1$ phases. The goal of Phase i for $i \in [0, \lfloor \log_{3/2}(n/m) \rfloor]$ is reached when all cells with representatives of cardinality at most $\lfloor m \cdot (3/2)^i \rfloor$ have been optimised. Each Phase i ends after a fixed number T_i of iterations, where T_i will be specified later. If Phase i does not achieve its goal within T_i iterations, we speak of a *failure* and will bound the failure probability of each phase.

Phase 0. The goal of Phase 0 is to optimise cells with representatives of cardinality at most m (which implies that all shortest paths of cardinality at most m were found). By the last statement of Theorem 4 with $k := m \ge \log n$, relying only on mutation steps, every cell M_{st} with representative of cardinality at most m contains a shortest path in expected time $O(\Delta^{3/4}n^{5/2}\log^{1/4} n)$ with probability $1 - o(1)$. Therefore, Phase 0 lasts only fails in $T_0 := O(\Delta^{3/4}n^{5/2}\log^{1/4}(n))$ iterations with probability $o(1)$.

Phase $i + 1$. Consider Phase $i + 1$ for $i \ge 0$ and assume that Phase i has concluded successfully. Then all cells with representatives of cardinality at most $\lfloor m \cdot (3/2)^i \rfloor$ have been optimised (which implies that all shortest paths of cardinality at most $\lfloor m \cdot (3/2)^i \rfloor$ have been found), while the optimisation of those with cardinality $k \in [\lfloor m \cdot (3/2)^i \rfloor + 1, \lfloor m \cdot (3/2)^{i+1} \rfloor]$ is underway. We refer to the latter cells as *target* cells. We divide Phase $i + 1$ into two subphases: a *crossover subphase*, followed by a *mutation subphase*. In the crossover subphase, we only rely on iterations with crossover, and the mutation subphase only relies on those that execute mutation. Let $g_i := (3/2)^{-i/6}m/4$, then the goal of the crossover subphase is that all target cells are g_i-near-complete. The goal of the mutation subphase is that all target cells contain complete shortest paths. We will show that each subphase concludes within in $O(\Delta^{3/4}n^{5/2}\log^{1/4}(n)(3/2)^{-i/6})$ iterations, with high probability. The total time spent in Phase $i + 1$ is thus $T_{i+1} = O(\Delta^{3/4}n^{5/2}\log^{1/4}(n)(3/2)^{-i/6})$.

Crossover subphase. Consider a target cell that is not yet g_i-near-complete and let $I = (v_0, \dots, v_k)$ be its representative solution. Then the following events suffice to make it g_i-near-complete. Crossover is executed (probability p_c), a first parent is chosen from a cell $M_{v_a v_j}$ with $a \in [0, g_i]$ and $j \in \mathcal{J} := [\max\{g_i, k - \lfloor m \cdot (3/2)^i \rfloor\}, \min\{k - g_i, \lfloor m \cdot (3/2)^i \rfloor\}]$ (probability $(g_i + 1)|\mathcal{J}|/(2n^2) \ge g_i|\mathcal{J}|/(2n^2)$) and a second parent is chosen from a cell $M_{v_j v_{k-b}}$ with $b \in [0, g_i]$ (probability $(g_i + 1)/n^2 \ge g_i/n^2$). If these events all happen, crossover concatenates paths encoded by the two parents. Since both sub-paths from v_a to v_j and from v_j to v_b have cardinality at most $m \cdot (3/2)^i$, the corresponding cells $M_{v_a v_j}$ and $M_{v_j v_{k-b}}$ are optimised, and thus the concatenated path from v_a to v_b is a shortest path. This implies that the original cell becomes g_i-near-complete. Note that $|\mathcal{J}|$ is non-increasing in k, thus it is minimised for $k = \lfloor m(3/2)^{i+1} \rfloor$ where it simplifies to

$$\begin{aligned}|\mathcal{J}| &\le |[\lfloor m(3/2)^{i+1} \rfloor - \lfloor m(3/2)^i \rfloor, \lfloor m(3/2)^i \rfloor]| - 2\lceil g_i + 1 \rceil\\ &= 2\lfloor m(3/2)^i \rfloor - \lfloor m(3/2)^{i+1} \rfloor + 1 - 2\lceil g_i + 1 \rceil\\ &\ge \lceil m(3/2)^i \rceil - 1 - m/2 - 4\end{aligned}$$

by Lemma 3 and using $\lceil g_i + 1 \rceil \le m/4 + 2$. We conclude that, in every iteration, the probability of an arbitrary but fixed target cell becoming g_i-near-complete is at least

$$p_{\text{cross}} := p_c \cdot \frac{g_i^2(\lceil m(3/2)^i \rceil - m/2 - 5)}{2n^4} = \Omega\left(\frac{m^3(3/2)^{\frac{2i}{3}}}{n^4}\right)$$

The probability of any fixed target cell not becoming g_i-near-complete within $3\ln(n)/p_{\text{cross}}$ iterations is at most $(1 - p_{\text{cross}})^{3\ln(n)/p_{\text{cross}}} \le e^{-3\ln n} = 1/n^3$ using $1 + x \le e^x$. By a union bound over at most n^2 target cells, the probability of not all target cells becoming g_i-near-complete in time $3\ln(n)/p_{\text{cross}} = O(n^4 m^{-3}\log(n)(3/2)^{-2i/3}) = O(\Delta^{3/4}n^{5/2}\log^{1/4}(n)(3/2)^{-i/6})$ is at most $1/n$.

Mutation subphase. Consider a target cell that is g_i-near-complete but not yet optimised and let $I = (v_0, \dots, v_k)$ be its representative solution. Let $a, b \le g_i$ be the smallest values such that the map contains a shortest path between v_a and v_{k-b}. If $a > 0$ then a shortest path between v_{a-1} and v_{k-b} is created if the following events occur. The algorithm applies mutation (probability $1 - p_c$), picks a solution of the optimised cell $M_{v_a v_b}$ as a parent (probability at least $1/n^2$), and decides to apply exactly one elementary operation (probability $1/e = \text{Pois}(1)$), where the edge (v_{a-1}, v_a) is added to the parent (probability at least $1/\Delta$). If $a = 0$ and $b > 0$ we use symmetric arguments to evolve a shortest path between v_a and v_{k-b+1}. In any case, the probability of extending optimised subpath of I by one edge in one iteration is at least $p_{\mathrm{mut}} := (1-p_c)/(\Delta n^2)$. We need at most $a+b \le 2g_i$ such events to optimise the shortest path between v_0 and v_k. Thus, the random time for this to happen is stochastically dominated by $Y := \sum_{j=1}^{2g_j} Y_j$ where the Y_j's are independent geometrically distributed random variables with success probability p_{mut}. Using $\mathbb{E}[Y] = 2g_i/p_{\mathrm{mut}}$, we obtain using Lemma 1 with $\lambda := \mathbb{E}[Y]$,

$$\begin{aligned}\Pr(Y \ge \mathbb{E}[Y] + \lambda) &\le e^{-\frac{1}{4}\min\left(\frac{\lambda^2}{2g_i/p_{\mathrm{mut}}^2}, \lambda p_{\mathrm{mut}}\right)} \\ &= e^{-\frac{1}{4}\min(2g_i, 2g_i)} = e^{-g_i/2}.\end{aligned}$$

Since $i \le \lfloor \log_{3/2}(n/m)\rfloor \le \log_{3/2}(n)$ we have $g_i = (3/2)^{-i/6} m/4 \ge n^{-1/6} m/4 = \Omega(\Delta^{-1/4} n^{1/3} \log^{1/4}(n)) = \Omega(n^{1/12})$ and thus the failure probability is $e^{-\Omega(n^{1/2})}$, that is, exponentially small. This still holds when taking a union bound over at most n^2 target cells. Hence, with high probability the mutation subphase is successful in $2\,\mathbb{E}[Y] = O(g_i/p_{\mathrm{mut}}) = O((3/2)^{-i/6} \Delta^{3/4} n^{5/2} \log^{1/4}(n))$ iterations.

Putting everything together. By a union bound over all failure probabilities in Phase 0 and all $O(\log n)$ Phases $i+1$, including both subphases, the probability of all phases being successful is at least $1 - o(1) - O(\log n)\cdot(1/n + e^{-\Omega(n^{1/2})}) = 1 - o(1)$. The total time is at most

$$T := \sum_{i=0}^{\lfloor \log_{3/2}(n/m)\rfloor} \left(\frac{3}{2}\right)^{-i/6} O(\Delta^{3/4} n^{5/2} \log^{1/4}(n))$$

which is $O(\Delta^{3/4} n^{5/2} \log^{1/4} n)$ as $\sum_{i=0}^{\infty}((3/2)^{-1/6})^i = \frac{1}{1-(3/2)^{-1/6}} = O(1)$. If a phase is unsuccessful, we repeat all phases and consider another period of T generations. Hence, the expected number of iterations to compute all shortest paths is at most $(1+o(1)) = O(T) = O(\Delta^{3/4} n^{5/2} \log^{1/4}(n))$. □

5 Speed-ups through Improved Selection

The bound from Theorem 7 is $O(n^{3.25} \log^{1/4} n)$ on graphs with $\Delta = \Omega(n)$. For graphs with large Δ, the runtime can be improved by adjusting the selection mechanism to only select feasible parents for crossover, thus eliminating idle steps. The idea is inspired by [Doerr *et al.*, 2013]. The first parent is selected from a cell M_{st} of M chosen uniformly at random. Then the second parent is selected uniformly at random from a cell in the row t but excluding column s (and of course excluding column t). If both cells are non-empty, the paths are concatenated and always form a valid path. Otherwise, no valid path is constructed and the algorithm skips directly to the next iteration.

Theorem 8. *The optimisation time of QD-GA with $p_c = \Omega(1)$ and using the above improved selection operator is $O(n^3 \log n)$ with probability $1 - o(1)$ and in expectation.*

Proof. We use the same notion of representative of a cell as in the proof of Theorem 7. The run of the algorithm is divided into $\lfloor \log_{3/2} \ell \rfloor \le \lfloor \log_{3/2} n \rfloor$ phases and a phase i ends when all cells with representative solutions of cardinality at most $\lfloor (3/2)^i \rfloor$ have been optimised. Note that phase 1 is completed at initialisation, thus we only look at phases $i+1$ for $i \ge 1$. During such a phase, all cells with representatives of cardinality at most $\lfloor (3/2)^i \rfloor$ have been optimised while the optimisation of those with cardinality $k \in [\lfloor (3/2)^i \rfloor + 1, \lfloor (3/2)^{i+1} \rfloor]$ is underway, and we refer to the latter cells as *target* cells.

Consider a target cell that is not yet optimised and let $I = (v_0, \dots, v_k)$ be its representative solution. For any integer $j \in [k - \lfloor (3/2)^i \rfloor, \lfloor (3/2)^i \rfloor]$ the paths $I_{v_0 v_j} = (v_0, \dots, v_j)$ and $I_{v_j v_k} = (v_j, \dots, v_k)$ are optimal solutions for cells $M_{v_0 v_j}$ and $M_{v_j v_k}$ respectively. Furthermore, those paths have the same cardinalities as the corresponding representatives of those cells, because otherwise I is not the representative of $M_{v_0 v_k}$ and this contradicts our assumption. These imply that $M_{v_0 v_j}$ and $M_{v_j v_k}$ have already been optimised since the cardinalities of $I_{v_0 v_j}$ and $I_{v_j v_k}$ are at most $\lfloor (3/2)^i \rfloor$. Thus picking solutions in those cells as parents and applying crossover (probability $p_c/(n(n-1)(n-2))$) optimises the cell $M_{v_0 v_k}$ by creating either solution I or an equivalently optimal solution. The number of such pairs of cells (or parents) is the number of possible integers j, which is at least

$$\begin{aligned}&\lfloor (3/2)^i \rfloor - (k - \lfloor (3/2)^i \rfloor) + 1 \\ &\ge 2\lfloor (3/2)^i \rfloor - \lfloor (3/2)^{i+1} \rfloor + 1 \ge \lceil (3/2)^i/2 \rceil - 1 =: \xi_i\end{aligned}$$

by Lemma 3. Thus, the number of iterations to optimise an arbitrary target cell in phase $i+1$ is stochastically dominated by a geometric random variable with parameter $p_c \xi_i / n^3 =: p_i$.

Applying Corollary 2 with $c = 9$ for this variable implies that with probability at most $n^{-9/4}$, an arbitrary target cell is not optimised after $\tau_i := (9 \ln n + 1) n^3/(p_c \xi_i)$ iterations. By a union bound on at most $n(n-1)$ target cells, the probability that the phase is not finished after τ_i iterations is at most $n(n-1)n^{-9/4} = n^{-1/4}$. Then by a union bound on at most $\lfloor \log_{3/2} n \rfloor$ phases the probability that all phases are not finished after $\tau := \sum_{i=1}^{\lfloor \log_{3/2} n \rfloor} \tau_i$ iterations is at most $n^{-1/4} \log_{3/2} n = o(1)$. Thus with probability $1 - o(1)$, all cells are optimised in time τ, and if this does not happen, we can repeat the argument, thus the expected running time of the algorithm is at most $(1+o(1))\tau$. Note that

$$\tau = O(n^3 \log n) \sum_{i=1}^{\lfloor \log_{3/2} n \rfloor} \xi_i^{-1} = O(n^3 \log n)$$

since $p_c = \Omega(1)$ and $\sum_{i=1}^{\lfloor \log_{3/2} n \rfloor} \xi_i^{-1} = O(1)$, which follows from $\xi_i^{-1} = O(1)$ for $i \le 4$ and $\xi_i \ge (3/2)^i/2 - 1 \ge (4/3)^i/2$ for $i \ge 5$, thus $\sum_{i=5}^{\infty} \xi^{-1} \le \frac{2}{1-(3/4)} = O(1)$. □

Combining the Theorems 4 and 8 gives the following result for the so-called fast QD-APSP algorithm.

Theorem 9. *QD-GA with constant $p_c \in (0,1)$ and with the improved selection operator, this setting is referred as the fast QD-APSP algorithm, has optimisation time $O(\min\{\Delta n^2 \cdot \max\{\ell, \log n\}, n^3 \log n\})$ on the APSP in expectation and with probability $1 - o(1)$.*

Fast QD-APSP takes the advantage of both operators, for example on balanced k-ary trees where $k = O(1)$ and $\ell = O(\log n)$, the expected running time of the algorithm is only $O(n^2 \log n)$. However, there also exist graphs where a tight optimisation time $\Omega(n^3 \log n)$ is required with high probability and also in expectation, even on a simple weight function that only assigns unit weights.

Theorem 10. *There is an instance where the Fast QD-APSP algorithm requires optimisation time $\Omega(n^3 \log n)$ with probability $1 - o(1)$ and in expectation.*

Proof. Let $n \geq 2$ be any natural number, we consider the graph $G = (V, E)$ with $|V| = 5n + 1$ which consists of:

- n paths $P_i = (u_i, c_i, v_i)$ of cardinality 2 where $i \in [n]$,
- a complete bipartite graph $K_{n,n}$ with two disjoint sets of nodes A, B where $|A| = |B| = n$, and for each pair $(a, b) \in A \times B$, we have $(a, b) \in E$ and $(b, a) \in E$,
- a node c, and the remaining edges of E are: (c, u_i) and (v_i, c) for each $i \in [n]$; (c_i, a), (b, c_i) for all $i \in [n]$, all $a \in A$, and all $b \in B$,

and the unit weight function $w\colon E \to \{1\}$. Figure 1 shows an example of this graph for $n = 2$. It is easy to see that G is strongly connected, and that its diameter is $\ell = 5 = O(1)$ (e. g. the cardinality of the shortest path from node u_i to v_j where $i \neq j$). The maximum degree is $\Delta = 3n = \Theta(|V|)$ which corresponds to the degree of a node in K_{2n}, but node c_i of any path P_i also has degree $2n + 2 = \Theta(|V|)$.

The unique shortest path between u_i and v_i is P_i, therefore we argue that with high probability Fast QD-APSP requires $\Omega(|V|^3 \log |V|) = \Omega(n^3 \log n)$ iterations to optimise the n cells $M_{u_i v_i}$ where $i \in [n]$. After initialization, none of these cells are optimised since P_i has cardinality 2. Then in every iteration, in order to optimise a cell $M_{u_i v_i}$ by crossover cells $M_{u_i c_i}$ and $M_{c_i v_i}$ must be selected as parents, and this occurs with probability $p_c/((5n+1)5n(5n-1))$. If mutation is used instead, i. e. with probability $1 - p_c$, either $M_{u_i c_i}$ and $M_{c_i v_i}$ cells need to be selected as the parent, i. e. with probability $2/((5n+1)5n)$, then one proper edge needs to be selected among $2n + 2$ of those that are connected to c_i

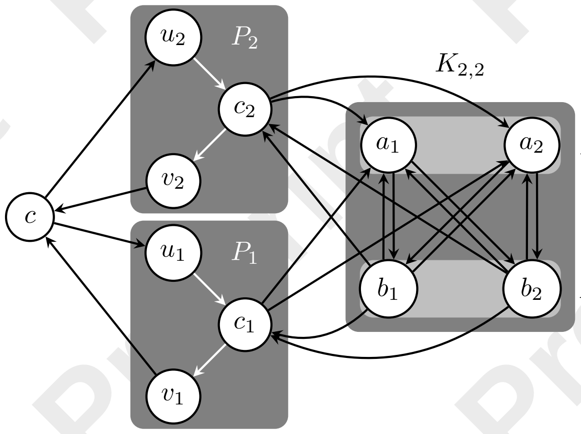

Figure 1: Graph G for $n = 2$, all edges have weight 1.

in an elementary operation to optimise $M_{u_i v_i}$, i. e. probability at most $1/(2n + 2)$. Note that it is possible to optimise $M_{u_i v_i}$ by first optimising some cell with a shortest path of larger cardinality and later mutating it back to $M_{u_i v_i}$. However, both shortening a shortest path that contains P_i as a subpath (e. g. converting path (u_i, c_i, v_i, c) to P_i) and mutating a shortest path that contains an edge of P_i (e. g. converting path (u_i, c_i, a) to P_i) require at least making the above correct elementary operation in the first place thus we can ignore these events. So overall, the probability of optimising $M_{u_i v_i}$ by either crossover or mutation is at most

$$2/((5n + 1)5n(2n + 2)) \leq 1/(25n^3).$$

This upper bound holds independently of i as far as the cell $M_{u_i v_i}$ has not yet been optimised.

Consider now the first $\tau := (1/5)(25n^3 - 1) \ln n$ iterations, the probability that an arbitrary cell $M_{u_i v_i}$ is not optimised during these iterations is at least, here using $(1 - 1/x)^{x-1} \geq 1/e$ for $x = 25n^3$, $\left(1 - \frac{1}{25n^3}\right)^{(1/5)(25n^3-1)\ln n} \geq n^{-1/5}$. Therefore, the probability that at least a cell among the n cells $M_{u_i v_i}$ is not optimised after $\tau = \Omega(n^3 \log n)$ iterations is at least $1 - (1 - n^{-1/5})^n \geq 1 - e^{-n^{-1/5} \cdot n} = 1 - o(1)$. Hence, Fast QD-APSP requires $\Omega(n^3 \log n)$ fitness evaluations to optimise G with probability $1 - o(1)$ and the expected runtime is $(1 - o(1)) \cdot \Omega(n^3 \log n) = \Omega(n^3 \log n)$. $\square$

6 Conclusions

Computing diverse sets of high quality solutions is important in various areas of artificial intelligence. Quality diversity algorithms have received a lot of attention in recent years due to their ability of tackling problems from a wide range of domains. We contributed to the theoretical understanding of these algorithms by providing the first analysis of a classical combinatorial optimisation problem that seeks multiple solutions in a natural behaviour space. Our analysis for the APSP has revealed several insights into the working behaviour of QD and the way that evolutionary operators exploit synergies between different regions of the behavioural space. We saw that mutation can extend shortest paths by appending edges, thus creating optimal paths for a different region of the behavioural space. Crossover exploits synergies much more effectively, as it concatenates two optimal paths to form an optimal path for a third region. Based on previous work by [Doerr and Theile, 2009], we also saw that the combination of crossover and mutation is particularly effective: while crossover quickly produces paths that are near-optimal, but have some gaps at one or both ends of a path, mutation can then fill these gaps by appending edges. Finally, we showed that improving the parent selection to always create feasible offspring during crossover gives a further speed-up on graphs with a large maximum degree.

Establishing a rigorous theoretical foundation and obtaining a better understanding of QD has the potential to improve practical applications across various domains. We hope that this work serves as a stepping stone towards the development of more efficient quality diversity algorithms and provides a basis for understanding QD algorithms for more complex planning problems.

Acknowledgments

This work has been supported by the Australian Research Council through grants DP190103894 and FT200100536.

References

[Baste *et al.*, 2019] Julien Baste, Lars Jaffke, Tomás Masarík, Geevarghese Philip, and Günter Rote. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019.

[Baste *et al.*, 2022] Julien Baste, Michael R. Fellows, Lars Jaffke, Tomás Masarík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A. Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022.

[Bossek and Neumann, 2022] Jakob Bossek and Frank Neumann. Exploring the feature space of TSP instances using quality diversity. In *Proc. of GECCO 2022*, pages 186–194. ACM, 2022.

[Bossek and Sudholt, 2024] Jakob Bossek and Dirk Sudholt. Runtime analysis of quality diversity algorithms. *Algorithmica*, 86(10):3252–3283, 2024.

[Corus and Lehre, 2018] Dogan Corus and Per Kristian Lehre. Theory driven design of efficient genetic algorithms for a classical graph problem. In *Recent Developments in Metaheuristics*, pages 125–140. Springer International Publishing, 2018.

[Covantes Osuna and Sudholt, 2022] Edgar Covantes Osuna and Dirk Sudholt. Runtime analysis of restricted tournament selection for bimodal optimisation. *Evolutionary Computation*, 30(1):1–26, 2022.

[Cully and Demiris, 2018] Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2018.

[Dang *et al.*, 2016] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima with diversity mechanisms and crossover. In *Proc. of GECCO 2016*, pages 645–652. ACM, 2016.

[Doerr and Neumann, 2020] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, 2020.

[Doerr and Theile, 2009] Benjamin Doerr and Madeleine Theile. Improved analysis methods for crossover-based algorithms. In *Proc. of GECCO 2009*, pages 247–254. ACM, 2009.

[Doerr *et al.*, 2012] Benjamin Doerr, Edda Happ, and Christian Klein. Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science*, 425:17–33, 2012.

[Doerr *et al.*, 2013] Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Frank Neumann, and Madeleine Theile. More effective crossover operators for the all-pairs shortest path problem. *Theoretical Computer Science*, 471:12–26, 2013.

[Floyd, 1962] Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM (CACM)*, 5(6):345, 1962.

[Fomin *et al.*, 2024] Fedor V. Fomin, Petr A. Golovach, Lars Jaffke, Geevarghese Philip, and Danil Sagunov. Diverse pairs of matchings. *Algorithmica*, 86(6):2026–2040, 2024.

[Fontaine *et al.*, 2020] Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proc. of GECCO 2020*, pages 94–102. ACM, 2020.

[Fontaine *et al.*, 2021] Matthew C. Fontaine, Ruilin Liu, Ahmed Khalifa, Jignesh Modi, Julian Togelius, Amy K. Hoover, and Stefanos Nikolaidis. Illuminating mario scenes in the latent space of a generative adversarial network. In *Proc. of AAAI 2021*, pages 5922–5930. AAAI Press, 2021.

[Friedrich *et al.*, 2009] Tobias Friedrich, Pietro S. Oliveto, Dirk Sudholt, and Carsten Witt. Analysis of Diversity-preserving Mechanisms for Global Exploration. *Evolutionary Computation*, 17(4):455–476, 2009.

[Gao and Neumann, 2014] Wanru Gao and Frank Neumann. Runtime analysis for maximizing population diversity in single-objective optimization. In *Proc. of GECCO 2014*, pages 777–784. ACM, 2014.

[Gao *et al.*, 2021] Wanru Gao, Samadhi Nallaperuma, and Frank Neumann. Feature-based diversity optimization for problem instance classification. *Evolutionary Computation*, 29(1):107–128, 2021.

[Gao *et al.*, 2022] Jie Gao, Mayank Goswami, Karthik C. S., Meng-Tsung Tsai, Shih-Yu Tsai, and Hao-Tsung Yang. Obtaining approximately optimal and diverse solutions via dispersion. In *Proc. of LATIN 2022*, volume 13568 of *LNCS*, pages 222–239. Springer, 2022.

[Gravina *et al.*, 2018] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. Fusing novelty and surprise for evolving robot morphologies. In *Proc. of GECCO 2018*, pages 93–100. ACM, 2018.

[Hagg *et al.*, 2018] Alexander Hagg, Alexander Asteroth, and Thomas Bäck. Prototype discovery using quality-diversity. In *Proc. of PPSN 2018*, volume 11101 of *LNCS*, pages 500–511. Springer, 2018.

[Hanaka *et al.*, 2021] Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, and Yota Otachi. Finding diverse trees, paths, and more. In *Proc. of AAAI 2021*, pages 3778–3786. AAAI Press, 2021.

[Hanaka *et al.*, 2023] Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In *Proc. of AAAI 2023*, pages 3968–3976. AAAI Press, 2023.

[Ingmar *et al.*, 2020] Linnea Ingmar, Maria Garcia de la Banda, Peter J. Stuckey, and Guido Tack. Modelling diversity of solutions. In *Proc. of AAAI 2020*, pages 1528–1535. AAAI Press, 2020.

[Jansen, 2013] Thomas Jansen. *Analyzing Evolutionary Algorithms – The Computer Science Perspective.* Springer, 2013.

[Johnson, 1977] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977.

[Lehman and Stanley, 2011] Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proc. of GECCO 2011*, pages 211–218. ACM, 2011.

[Medina *et al.*, 2023] Alejandro Medina, Melanie Richey, Mark Mueller, and Jacob Schrum. Evolving flying machines in minecraft using quality diversity. In *Proc. of GECCO 2023*, pages 1418–1426. ACM, 2023.

[Miao *et al.*, 2022] Jiayu Miao, Tianze Zhou, Kun Shao, Ming Zhou, Weinan Zhang, Jianye Hao, Yong Yu, and Jun Wang. Promoting quality and diversity in population-based reinforcement learning via hierarchical trajectory space exploration. In *Proc. of ICRA 2022*, pages 7544–7550. IEEE, 2022.

[Mouret and Clune, 2015] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015.

[Neumann and Witt, 2010] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[Neumann *et al.*, 2018] Aneta Neumann, Wanru Gao, Carola Doerr, Frank Neumann, and Markus Wagner. Discrepancy-based evolutionary diversity optimization. In *Proc. of GECCO 2018*, pages 991–998. ACM, 2018.

[Neumann *et al.*, 2019] Aneta Neumann, Wanru Gao, Markus Wagner, and Frank Neumann. Evolutionary diversity optimization using multi-objective indicators. In *Proc. of GECCO 2019*, pages 837–845. ACM, 2019.

[Nikfarjam *et al.*, 2022] Adel Nikfarjam, Anh Viet Do, and Frank Neumann. Analysis of quality diversity algorithms for the knapsack problem. In *Proc. of PPSN 2022*, volume 13399 of *LNCS*, pages 413–427. Springer, 2022.

[Nikfarjam *et al.*, 2024a] Adel Nikfarjam, Aneta Neumann, and Frank Neumann. On the use of quality diversity algorithms for the travelling thief problem. *ACM Transactions on Evolutionary Learning and Optimization*, 4(2):12, 2024.

[Nikfarjam *et al.*, 2024b] Adel Nikfarjam, Ty Stanford, Aneta Neumann, Dorothea Dumuid, and Frank Neumann. Quality diversity approaches for time-use optimisation to improve health outcomes. In *Proc. of GECCO 2024*, page 1318–1326. ACM, 2024.

[Pugh *et al.*, 2016] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Searching for quality diversity when diversity is unaligned with quality. In *Proc. of PPSN 2016*, volume 9921 of *LNCS*, pages 880–889. Springer, 2016.

[Qian *et al.*, 2024] Chao Qian, Ke Xue, and Ren-Jian Wang. Quality-diversity algorithms can provably be helpful for optimization. In *Proc. of IJCAI 2024*. ijcai.org, 2024.

[Ren *et al.*, 2024] Shengjie Ren, Zhijia Qiu, Chao Bian, Miqing Li, and Chao Qian. Maintaining diversity provably helps in evolutionary multimodal optimization. In *Proc. of IJCAI 2024*, pages 7012–7020. ijcai.org, 2024.

[Schmidbauer *et al.*, 2024] Marcus Schmidbauer, Andre Opris, Jakob Bossek, Frank Neumann, and Dirk Sudholt. Guiding quality diversity on monotone submodular functions: Customising the feature space by adding Boolean conjunctions. In *Proc. of GECCO 2024*, page 1614–1622. ACM, 2024.

[Shen *et al.*, 2020] Ruimin Shen, Yan Zheng, Jianye Hao, Zhaopeng Meng, Yingfeng Chen, Changjie Fan, and Yang Liu. Generating behavior-diverse game AIs with evolutionary multi-objective deep reinforcement learning. In *Proc. of IJCAI 2020*, pages 3371–3377. ijcai.org, 2020.

[Sudholt and Thyssen, 2012] Dirk Sudholt and Christian Thyssen. Running time analysis of ant colony optimization for shortest path problems. *Journal of Discrete Algorithms*, 10:165–180, 2012.

[Sudholt, 2020] Dirk Sudholt. The benefits of population diversity in evolutionary algorithms: A survey of rigorous runtime analyses. In *Theory of Evolutionary Computation - Recent Developments in Discrete Optimization*, Natural Computing Series, pages 359–404. Springer, 2020.

[Vonásek and Saska, 2018] Vojtech Vonásek and Martin Saska. Increasing diversity of solutions in sampling-based path planning. In *Proc. of ICRAI 2018*, pages 97–102. ACM, 2018.

[Witt, 2014] Carsten Witt. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processing Letters*, 114(1-2):38–41, 2014.

[Zardini *et al.*, 2021] Enrico Zardini, Davide Zappetti, Davide Zambrano, Giovanni Iacca, and Dario Floreano. Seeking quality diversity in evolutionary co-design of morphology and control of soft tensegrity modular robots. In *Proc. of GECCO 2021*, pages 189–197. ACM, 2021.