A Genetic Algorithms Approach for Inverse Shortest Path Length Problems

António Leitão

University of Coimbra

Adriano Vinhas

University of Coimbra

Penousal Machado

University of Coimbra

Francisco Câmara Pereira

University of Coimbra

Authors Note

António Leitão, CISUC, Department of Informatics Engineering,
University of Coimbra, 3030-290 Coimbra, Portugal
Tel.: +351-239-790-000, Fax: +351-239-701-266
E-mail: apleitao@dei.uc.pt

Adriano Vinhas, CISUC, Department of Informatics Engineering
University of Coimbra, 3030-290 Coimbra, Portugal
Tel.: +351-239-790-000, Fax: +351-239-701-266
E-mail: avinhas@student.dei.uc.pt

Penousal Machado, CISUC, Department of Informatics Engineering
University of Coimbra, 3030-290 Coimbra, Portugal
Tel.: +351-239-790-000, Fax: +351-239-701-266
E-mail: machado@dei.uc.pt

Francisco Câmara Pereira, Department of Informatics Engineering
University of Coimbra
Singapore-MIT Alliance for Research and Technology
S16-05-08, 3 Science Drive 2
Singapore 117543, Singapore
E-mail: camara@smart.mit.edu

Abstract

Inverse Combinatorial Optimization has become a relevant research subject over the past decades. In graph theory, the Inverse Shortest Path Length problem becomes relevant when we don't have access to the real cost of the arcs and want to infer their value so that the system has a specific outcome, such as one or more shortest paths between nodes. Several approaches have been proposed to tackle this problem, relying on different methods, and several applications have been suggested. This study explores an innovative evolutionary approach relying on a genetic algorithm. Two scenarios and corresponding representations are presented and experiments are conducted to test how they react to different graph characteristics and parameters. Their behaviour and differences are thoroughly discussed. The outcome supports that evolutionary algorithms may be a viable venue to tackle Inverse Shortest Path problems.

Keywords: Inverse Combinatorial Optimization, Inverse Shortest Path Length, Genetic Algorithms

Introduction

Graph theory is highly appraised in Computer Science as well as several other fields, with applications on a large number of well documented problems (Gross & Yellen, 2005). A formal definition of a graph has been given for instance by Gross and Yellen (2005), but as succinctly as possible, a graph is a collection of nodes connected (or partially connected) by arcs with an associated cost such that successive arcs form paths between nodes. There are many widely known problems in the literature related to graph theory, such as that of finding the shortest path between two nodes (source and sink). This problem has been widely studied in the past and several efficient solutions have been developed by the scientific community (Bellman, 1956; Bertsekas, 1991; Dijkstra, 1959; Floyd, 1962; Johnson, 1977; Moore, 1959).

However, when solving such optimization problems, we often assume that we have access to all the information required and that this data is accurate. Often in real applications this is not the case. In shortest path problems, for instance, we may not have access to the real cost of the arcs (distances, travel times, etc.), we may have estimates or only partial knowledge of their values. A possible solution to extend our knowledge on the arc costs is to consider the inverse problem. This requires only that we have access to the topology of the graph and a desired outcome for the system, such as one or more shortest paths between pairs of $(source, sink)$ nodes. Inverse optimization allows a vector of arc costs to be calculated, or at least estimated, so that our desired outcome is optimal and the path costs as close as possible to previous estimations that we may have access to.

Inverse optimization problems are highly complex and widely relevant, but so far have failed to attract the attention of the Evolutionary Computation community. The need to test the viability of evolutionary approaches to tackle inverse optimization problems or in this case Inverse Shortest Path Length (ISPL) problems specifically is the main motivation for this study. We propose an unprecedented Genetic Algorithm approach which is tested on two different scenarios

and hope to provide a baseline for future work on the subject. We experiment with several design choices, compare the obtained results, and discuss the resulting behaviours. The following section covers Inverse Combinatorial Optimization, previous work on both Inverse Solution Optimization Problems and Inverse Objective Value Optimization Problems is presented and discussed. Afterwards, the ISPL problem is introduced and our Genetic Algorithms approach described. Our experimentation setup, including the applied methodology, is described and the obtained results are presented and analysed. Finally, conclusions are drawn and future work proposed.

Inverse Combinatorial Optimization

On a broad analysis, inverse optimization problems can be branched into two classes: inverse solution optimization problems and inverse objective value optimization problems (Hung & Director-Sokol, 2003). Briefly, inverse solution optimization problems require finding an arc costs vector that makes a desired solution optimal. Variations include the minimum cut problem (Zhang & Cai, 1998), inverse maximum flow problem (Yang, Zhang, & Ma, 1997), inverse center location problem (Cai, Yang, & Zhang, 1999), the inverse shortest path problem (ISPP) (Cai & Yang, 1994), among others. Inverse objective value optimization problems require that a solution is found that matches a desired value for the objective function rather than assuming there is a desired optimal solution. This type of problem has been much less explored than inverse solution optimization problems but the reverse center location (Zhang, Yang, & Cai, 1999) and the Inverse Shortest Path Length (ISPL) (Fekete, Hochstättler, Kromberg, & Moll, 1999) are examples, with ISPL probably being the most widely known. Heuberger (2004) presented various methods that had been successfully applied in the past to a set of inverse combinatorial problems on an extended survey. The following subsections further explore the two classes of inverse combinatorial optimization and previous work regarding each one.

Inverse Solution Optimization Problems

Burton and Toint (1992) were likely the first to address the ISPP. They present a

specialization of the dual Quadratic Programming method by Goldfarb and Idnani (1983), which computes a sequence of optimal solutions based on constraints. These constraints are incorporated into the algorithm at each step and contribute to reaching the desired optimum. Zhang et al. (1995) considered the problem as a Linear Programming model rather than a Quadratic Programming one and transformed it into a convenient form. They then proposed a column generation algorithm for the Linear Programming model which is arguably more effective to handle large size problems.

Xu and Zhang (1995) have addressed the ISPP, looking for weighted vectors under which the shortest paths correspond to the desired ones. They suggest that such solution vectors form a polyhedral cone and show a close relation between the ISPP and the minimal cutset problem in graph theory. Zhang and Cai (1998) have studied a general inverse minimum cut problem with multiple cuts and bounded increments and decrements. They also attribute specific costs to adaptations on the capacity of each arc. It is shown that this problem can be transformed into a minimum cost circulation problem and is therefore solvable efficiently by polynomial algorithms. The minimum cut problem as well as the inverse maximum flow problems have also been addressed by Yang et al. (1997) and the ISPP by Cai and Yang (1994).

Cai et al. (1999) have discussed the inverse center location problem. They question whether the inverse problem is solvable in polynomial time whenever the original is. They show that the inverse center location problem contradicts this hypothesis as it is NP-hard despite its counterpart having a strongly polynomial method to solve.

Mao-Cheng and Li (1997) consider the inverse matroid intersection problem which, as described, is a generalization of several Inverse Solution Optimization Problems, such as the ISPP, the inverse minimum spanning tree, the inverse bipartite matching and the inverse minimum arborescence problems. The inverse matroid intersection problem is formulated as a combinatorial linear program but may also be transformed into a minimum cost circulation problem. The study is finally extended into approaches with multiple common independent sets.

Liu and Zhang (2003) review part of the existing methods applied to Inverse Solution Optimization Problems into four categories: simplex methods for general inverse linear programming problems, combinatorial algorithms for certain special inverse combinatorial optimization problems, column generating methods and ellipsoid methods for specific inverse combinatorial linear programming problems. They address the maximum-weight problem in nonbipartite graphs and propose a binary search algorithm as well as an ascending algorithm and an ellipsoid method.

Day et al. (2002) address the Inverse Multicommodity Flow Problem as a way to find the appropriate set of costs for a railroad system for shortest-path routing with impedances. As the problem is too large to be feasible, Day (2002) proposes preprocessing methods that ensure its feasibility while transforming the problem as little as possible. The authors propose a primal-dual solution algorithm and a hot start basis-finding algorithm to tackle the problem within a reasonable amount of time. They also consider using Lagrangian relaxation.

Ahuja and Orlin (2001) propose a unified approach to several inversion problems. They develop a unified framework which they first apply to the inverse linear programming problem and then specialize on the ISPP, the assignment problem, the minimum cut problem and the minimum cost flow problem, achieving faster algorithms for each. They report improvements over previous results on each algorithm and also produce new results on inverse linear programming and inverse network flow models. Other studies by the same research group are also available (Ahuja & Orlin, 2000; Ahuja & Orlin, 2001; Ahuja & Orlin, 2002; Sokkalingam, Ahuja, & Orlin, 1999). Hochbaum (2003)tackled the inverse spanning-tree problem, relying on a formulation different from those used by Sokkalingam et al. (1999) and Ahuja and Orlin (2000). He exploits the structure of the graph that results from his formulation to apply less constraints and therefore improve algorithm complexity.

Inverse Objective Value Optimization Problems

Ahmed and Guan (2004) consider the inverse optimal value problem where they originally

have a linear program with a set of cost vectors restricted to a convex compact set. In their study they show it to be a NP-hard task unless under certain structural characterizations that reduce the problem to a concave maximization or concave minimization problem. They also describe an algorithm based on solving linear and bilinear programming models.

Fakete et al. (1999) have studied ISPL problems focusing on the characteristics of the graphs and the idea of estimating travel times in road networks based on a set of source-destination pairs. They consider a general case and show it to be NP-complete. They also consider specific cases such as planar graphs, directed graphs and special distance graphs. Certain polynomially solvable cases are also identified.

Hung and Sokol (2003) have conducted an extensive study on ISPL problems. They report ISPL to be NP-complete on the general case and addresses several complex cases in order to find out when it becomes intractable. Polynomially solvable cases are also discussed. They propose a set of heuristics as well, discuss their characteristics and tests their performance on several problems. Finally they apply these heuristics to datasets from real world telecommunications and report worse results than on random instances due to infeasibility issues on real world instances.

Burton, Pulleyblank and Toint (1997) examine the case where upper bounds are set on ISPPs. This expansion on their aforementioned work is classified as an Inverse Objective Value Optimization Problem study as several solutions may be accepted as long as the defined constraints are met. They show that the problem is NP-Complete and test a local minimisation algorithm on several examples, while analysing its behaviour.

Zhang et al. (1999) have tackled a generalised reverse center location problem where distances between given pairs of vertices are bounded by predefined limits. The authors consider this problem as a special case of that studied by Burton et al. (1997). They differ in the objective function which in this case can manage different cost coefficients on various arcs, while restrictions on the magnitude of adjustments are applied as well. They formulated the problem as a mixed

integer programming model which is relaxed to find approximate global optimal solutions.

The same research group later expanded on this study by addressing a special case (Zhang, Yang, & Cai, 2004) and not only by considering the original problem under different conditions but also by considering a vertex-to- points approach, where the distance from a vertex to all the points on the edges are maintained below a given upper bound (Zhang, Yang, & Cai, 2004 (2)). Zhang and Lin have also shown that the general model is NP-complete and have proposed a dynamic programming model for the special case of a single $(source, sink)$ pair and a combinatorial algorithm for the case of trees with a single source.

Cui and Hochbaum (2010) focus on special cases of the ISPL, such as the $(single\ source, all\ sink)$ problem or the $(single\ source, single\ sink)$ problem. While these scenarios can be solved in polynomial time for the Feasible ISPL variance, it is shown that they are NP-complete for the general ISPL formulation. The study also identifies cases where ISPL becomes polynomially solvable and addresses the Lower Bounding ISPL, where shortest path lengths are kept above given bounds.

The Inverse Shortest Path Length Problem

In this section the ISPL problem is formally described. We try to maintain the notation used by Cui and Hochbaum (2010) as much as possible. Then, a genetic algorithms approach to the ISPL is introduced.

Formal Statement

An instance of the ISPL problem can be described using two graphs over the same set of nodes V and arcs A, which in these case are undirected, without loss of generalization. The graph $G = (V, A)$ includes a nonnegative weight vector c which lists our original estimations for the weight of each arc. G includes therefore the paths between each $(source, sink)$ pair in V. A second graph $G_d(V, A_d)$ describes the pairwise distance $d_{ij} \geq 0$ which correspond to the desired shortest path distance between nodes i, j for each $(source, sink)$ pair in V. G_d includes therefore a distances

vector d. The first graph is often referred to as the *network graph* and the second one as the *distances graph*.

An ISPL problem instance can therefore be defined as $G(c) = (V, A, c)$ and $G_d(d) = (V, A_d, d)$. Furthermore, let $P_{st}(x)$ denote the shortest path from s to t in $G(x)$, $D_{st}(x)$ be the length of $P_{st}(x)$ and $D(P, x)$ contain the length of path P in $G(x)$.

A penalty function is usually also required to act on deviations on arc weights in G. For each arc $(i, j) \in A$ there is a function $f_{ij}: \Re \to \Re^+$ where $f_{ij}(x_{ij} - c_{ij})$ is the cost of adjusting the cost of arc (i, j) from c_{ij} to x_{ij}. Various functions can be applied here such as monotonically decreasing or monotonically increasing functions (Cui & Hochbaum, 2010). Usually, the penalty function satisfies $f_{ij}(0) = 0$, for all $(i, j) \in A$. In this study $f_{ij}(x_{ij} - c_{ij}) = |x_{ij} - c_{ij}|$.

Considering $G(c)$ and $G_d(d)$, the weight vector $x: A \to \Re_+^{|A|}$ is said to be G_d *satisfying* if $D_{st}(x) = d_{st}$ for all paths $(s, t) \in A_d$. Therefore, the ISPL consists on finding a G_d *satisfying* weight vector x on G that minimizes the penalty cost $F(x)$ which is the sum of the penalty functions for all arcs, $F(x) = \sum_{(i,j) \in A} f_{ij}(x_{ij} - c_{ij})$.

The definition above or equivalent ones are broadly used on existing studies in the literature, there are however variations such as the Feasible ISPL, the Lower Bounding ISPL or the Upper Bounding ISPL. The approach followed in this study is however the original ISPL formulation, as described above.

A Genetic Algorithms Approach

This section describes how the ISPL problem was tackled using genetic inspired approach, focusing on design choices regarding the GA. To develop our setup and run experiments we opted to use the ECJ simulator (Luke, et al., 2001). This simulator allows a broad control over the design aspects of the algorithm such as representations, selection, crossover and mutation operators as well as associated probabilities and other relevant parameters.

The algorithm relies on a population of solution candidates which, by means of different

operators, searches for increasingly better solution candidates to hopefully solve the task. Each individual of the population represents a costs vector$c$as a chromosome, which is easily mapped to a $G(c)$ graph with the aid of a stored G graph. In detail, each chromosome contains a set of genes, each gene representing the cost of an individual arc in G. Therefore, the size of the chromosome corresponds to the number of arcs in G.

Regarding operators, we opted for selecting parents through tournament selection, they undergo two-point crossover and the resulting offspring are then subjected to gaussian mutation. Since gaussian mutation is being applied, it is important to analyse the impact of varying standard deviation values associated with the distribution. Also, elitism is applied, meaning that the best individual from each generation is maintained to the next one.

The fitness function used in the GA has been partially discussed in the previous section. In detail, a $G_d(d)$ graph is stored and used in the evaluation process of each individual. During this evaluation process, for each pair (s, t), the distance D_{st} is calculated based on Dijkstra's algorithm (Dijkstra, 1959) and compared to the prescribed value d_{st} through $F(x) = \sum_{(i,j) \in A} f_{ij} \left(\left| x_{ij} - c_{ij} \right| \right)$ using the same monotonically increasing function within the interval $]0, \infty[$. The penalty cost is therefore able to assess the quality of each individual of the population and corresponds to the fitness function applied during evaluation. Obviously, we're dealing with a minimization problem, meaning that selection will favour individuals whose fitness values are closer to 0 over larger values.

Two different scenarios are considered: the first one assumes that we have no arc cost estimations; the second one uses initial arc cost estimations. These two scenarios allow us to address two different situations that may come up on real world problems. We expect the algorithm to achieve better results on the second scenario than on the first one as we only have to evolve arc cost adjustments rather than start searching for adequate costs from a blind estimation.

On the first scenario, each gene of an individual corresponds directly to an arc cost and its

initial value is sampled from a uniform distribution within the interval [0,1]. On the second scenario, initial cost estimations are established. Each gene of an individual corresponds to an adjustment to be made to the corresponding initial estimation rather than directly to an arc cost. For this reason, initial gene values are sampled from a gaussian distribution with $\mu = 0$ and $\sigma = 0.15$ which seems appropriate as we want adjustments to start at small values and evolve from there. Gene values are contained in the interval $[-1,1]$ and arc costs in $[0,1]$. During evaluation, in order to calculate an arc cost, x_{ij}, we add adjustments to initial cost estimations and obtain a vector c.

In both scenarios, constraints are set on the mutation operator so that gene values are maintained inside their appropriate intervals. In detail, genes are disturbed using values sampled repeatedly from a gaussian distribution with $\mu = 0$ until a valid result comes up or until a maximum number of attempts is completed ($max = 5$), at which point no mutation occurs. Various σ values are tested. Also during evaluation on the second scenario, cost values are truncated to the appropriate interval.

Experimentation

This section explains the experimental setup used to test our GA on the ISPL problem. Firstly, the methodology is assessed, including a parametric study devised to find the most adequate parameters, and a description of our final experiments. The results are then reported and analysed.

Methodology

The relevant parameters concerning the GA are described in Table 1. A study was conducted on the mutation's probability and standard deviation and how they affect the algorithm's behaviour. This study included both scenarios. Table 1 describes the tested parameters. The column labeled range of values shows the tested values for the assessed parameters and the column labeled default value shows fixed values. When a parameter is being studied, all other are attributed the default value.

Concerning the generation of graphs, several graph structures were designed, each with

different characteristics thus allowing us to explore the behaviour of the GA under different conditions. For this purpose, the *networkx* package (Hagberg, Swart, & S Chult, 2008) was used for the generation of random graphs, although the source code was modified in order to allow us to easily specify the number of arcs desired for each graph. Notice that in this study a graph can't have multiple arcs with the same source and target and a maximum number of arcs for each graph was established. This means that a graph with n nodes can have at most $n \times (n-1)/2$ arcs.

Table 1: Parameter study (related to the genetic algorithm) used in the assessment of the algorithm.

Parameter	Range of values	Default Value
Number of generations	-	100
Number of individuals	-	100
Tournament size	-	5
Crossover Probability	-	0.8
Mutation Probability	[0.01, 0.02, 0.05, 0.10]	0.05
Mutation Std. Deviation	[0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30]	0.1
Source-Sink Disturb	[0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30]	0.1

The tests were organised in 4 different setups. Setups 1 and 2 aim to test the algorithm's behaviour using smaller graphs while setups 3 and 4 test the algorithm's performance with bigger graphs. Table 2 includes all the tests completed for each setup as well as tested graph parameters. Notice that on setups 1 and 2, for each tested graph, 20 repetitions were completed. On setups 3 and 4, due to large computational effort requirements, only 10 repetitions were completed for each graph tested.

Besides having to generate graph structures for G, we also needed to generate the associated G_d graph with the desired shortest path distances. This set was built based on a predefined number of $(source, sink)$ paths. In order to do so we attribute weights to the graph using a uniform distribution within the interval $[0,1]$ and save the shortest paths between random $(source, sink)$ pairs using a Dijkstra's algorithm (Dijkstra, 1959). The generated G and G_d graphs are maintained and used in each repetition.

Finally, the desired shortest path costs are disturbed, resulting on a smaller chance that a viable set of desired shortest path costs exists as they may conflict with each other. This approach makes the ISPL problem much tougher to solve and reduces the chances that a viable solution can be found by the algorithm. In detail, this is accomplished by disturbing the cost of each arc belonging to a shortest path on the generated graph using values sampled from an initial distribution with $\mu = 0$ (different σ values are tested). Only after disturbing the costs, are the desired shortest path costs determined on both scenarios. Specifically on the second scenario, the initial estimated costs of the arcs are set during the generation of the graphs, before disturbing their value. Therefore, the estimated costs are an approximation of the true arc costs, which become worse and worse estimations as the value of σ is increased, thus making it an increasingly tougher problem to solve.

Table 2: *Sets of Graphs used for the assessment.*

Configuration	Graphs	Repetitions	Nodes	Arcs	Paths	Description
Setup 1	30	20	15	100	100	Subset1
			30	100	100	Subset2
			30	200	100	Subset3
Setup 2	30	20	15	100	200	Subset1
			30	100	200	Subset2
			30	200	200	Subset3
Setup 3	3	10	50	500	1000	Subset1
			100	500	1000	Subset2
			50	1000	1000	Subset3
			100	1000	1000	Subset4
Setup 4	3	10	50	500	3000	Subset1
			100	500	3000	Subset2
			50	1000	3000	Subset3
			100	1000	3000	Subset4

The parametric study was conducted for each parameter individually, maintaining all the others at their default value. Only the smaller graphs, those from setups 1 and 2, were used during these study. For the graphs in setups 3 and 4, the parameters that we considered best were applied.

The results from these tests are shown and discussed in the following section.

Results and Analysis

This subsection includes a discussion of the obtained results for the tested setups. As previously mentioned, a parametric study was conducted on setups 1 and 2, whose results are firstly addressed. The discussion starts by focusing on the effect of different mutation probabilities (Tables 3 and 4), followed by mutation standard deviations (Tables 5 and 6) and finally by the effect of disturbance on shortest path costs (Tables 7 and 8). A discussion on how the algorithm reacts to varying graph structures as well as to very large graphs is then presented.

$$MBF = \frac{\sum_{i=1}^{n} \frac{bestfitness}{avg(c_{ij}) \times p}}{n} \quad (1)$$

$$MAF = \frac{\sum_{i=1}^{n} \frac{averagefitness}{avg(c_{ij}) \times p}}{n} \quad (2)$$

The results shown on the aforementioned tables are based on the fitness values obtained on all repetition sets ran on the tested graphs. However, different graphs correspond to different desired shortest path sets which may present different challenges as they surely have different lengths and costs. Also, different setups and subsets use graphs with different characteristics, which also influence the difficulty of the problem. Therefore, different experiments can't be properly compared using the fitness values. In this case, the Mean Best Fitness (MBF) and Mean Average Fitness (MAF) values shown in each table are obtained by Equations 1 and 2 respectively, where $n = num\ graphs \times repetitions$ and p is the number of paths. A Wilcoxon Mann Whitney test with a significance level of 0.01 was conducted, comparing the results obtained on each scenario. The instances where the algorithm performed significantly better were presented in bold in all tables.

Table 3: Results obtained on both scenarios during the mutation probability study for setup 1. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.

Scenario	Mut. Prob.	Subset 1		Subset 2		Subset 3	
		MAF	MBF	MAF	MBF	MAF	MBF
1	0.01	0.3263	0.3111	0.2938	0.2782	0.5377	0.5155
	0.02	0.2622	0.2439	0.2212	0.2059	0.3984	0.3741
	0.05	0.2149	0.1892	0.1615	0.1431	0.2737	0.2439
	0.10	0.1961	0.1611	0.1480	0.1220	0.2416	0.2027
2	0.01	0.1185	**0.1045**	0.0580	**0.0547**	0.1001	**0.0939**
	0.02	0.1241	**0.1055**	0.0622	**0.0554**	0.1087	**0.0963**
	0.05	0.1291	**0.1065**	0.0813	**0.0642**	0.1416	**0.1132**
	0.10	0.1896	0.1439	0.1085	**0.0790**	0.1843	**0.1383**

Table 4: Results obtained on both scenarios during the mutation probability study for setup 2. MBF stands for the mean best fitness and MAF for the mean average fitness obtained onall repetitions and graphs.

Scenario	Mut. Prob.	Subset 1		Subset 2		Subset 3	
		MAF	MBF	MAF	MBF	MAF	MBF
1	0.01	0.3696	0.3538	0.3022	0.2881	0.5755	0.5537
	0.02	0.2899	0.2729	0.2323	0.2186	0.4375	0.4150
	0.05	0.2333	0.2111	0.1742	0.1579	0.3130	0.2872
	0.10	0.2251	0.1933	0.1612	0.1380	0.2780	0.2449
2	0.01	0.1291	**0.1261**	0.0721	**0.0692**	0.1359	**0.1304**
	0.02	0.1337	**0.1273**	0.0759	**0.0701**	0.1423	**0.1315**
	0.05	0.1525	**0.1338**	0.0932	**0.0782**	0.1737	**0.1490**
	0.10	0.1848	**0.1491**	0.1200	**0.0932**	0.2147	**0.1740**

Tables 3 and 4 show the results obtained on both scenarios, on setups 1 and 2 respectively, using various mutation probabilities. While the results shown in Table 4 appear to be worse than those shown in Table 3 the reaction to increasing mutation probabilities are consistent for all subsets. Interestingly, when comparing both scenarios, the reactions are contrasting. On scenario 1, the algorithm gets advantages from using a larger mutation probability, while on scenario 2 it benefits from a decreasing probability. This contrasting reaction is most likely the result of using estimates on the second scenario.

Table 5: Results obtained on both scenarios during the mutation standard deviation study for setup 1. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.

Scenario	Mut. St. Dev.	Subset 1		Subset 2		Subset 3	
		MAF	MBF	MAF	MBF	MAF	MBF
1	0.01	0.3110	0.3062	0.2742	0.2705	0.4231	0.4170
	0.05	0.2406	0.2249	0.2108	0.1986	0.3489	0.3285
	0.10	0.2149	0.1892	0.1615	0.1431	0.2737	0.2439
	0.15	0.1855	0.1554	0.1459	0.1215	0.2416	0.2050
	0.20	0.1887	0.1518	0.1493	0.1183	0.2362	0.1927
	0.25	0.1966	0.1536	0.1574	0.1204	0.2431	0.1930
	0.30	0.2050	0.1566	0.1668	0.1243	0.2526	0.1967
2	0.01	0.0987	**0.0971**	0.0622	**0.0606**	0.1091	**0.1062**
	0.05	0.1014	**0.0929**	0.0630	**0.0551**	0.1102	**0.0963**
	0.10	0.1291	**0.1065**	0.0813	**0.0642**	0.1416	**0.1132**
	0.15	0.1366	**0.1069**	0.1009	**0.0746**	0.1734	**0.1313**
	0.20	0.1532	**0.1146**	0.1185	**0.0836**	0.2006	**0.1468**
	0.25	0.1696	**0.1226**	0.1336	**0.0913**	0.2252	**0.1604**
	0.30	0.1833	0.1287	0.1477	**0.0982**	0.2455	**0.1715**

On scenario 1, individuals are initialized through a uniform distribution and initial gene values are only blind estimates for arc costs, meaning that the algorithm will have to search through the entire search space of arc costs. For this reason, it is likely to benefit from a larger number of mutations in order to get closer to an optimal solution. On the other hand, on scenario 2, individuals are initialized through a gaussian distribution with $\mu = 0$ and $\sigma = 0.15$ and each gene represents a cost disturbance to be added to estimated initial values. In this case, desired shortest path costs are disturbed using a gaussian distribution with $\mu = 0$ and $\sigma = 0.10$ as it is the default value, meaning that the estimated costs are not likely to be too far away from optimal (or as close as possible) values. Therefore, even though this scenario results on a larger search space than on scenario 1, considering the above mentioned initialization and disturbance setups, the algorithm benefits from smaller number of mutations to attempt reaching an optimal solution on all subsets.

Table 6: Results obtained on both scenarios during the mutation standard deviation study for setup 2. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.

Scenario	Mut. St. Dev.	Subset 1		Subset 2		Subset 3	
		MAF	MBF	MAF	MBF	MAF	MBF
1	0.01	0.3357	0.3313	0.2962	0.2927	0.4547	0.4489
	0.05	0.2706	0.2564	0.2280	0.2171	0.3920	0.3739
	0.10	0.2333	0.2111	0.1742	0.1579	0.3130	0.2872
	0.15	0.2071	0.1804	0.1626	0.1401	0.2805	0.2494
	0.20	0.2094	0.1767	0.1632	0.1346	0.2763	0.2386
	0.25	0.2159	0.1772	0.1632	0.1373	0.2812	0.2378
	0.30	0.2250	0.1808	0.1806	0.1411	0.2901	0.2411
2	0.01	0.1270	**0.1256**	0.0765	**0.0752**	0.1434	**0.1411**
	0.05	0.1293	**0.1218**	0.0758	**0.0693**	0.1427	**0.1314**
	0.10	0.1525	**0.1338**	0.0932	**0.0782**	0.1737	**0.1490**
	0.15	0.1656	**0.1372**	0.1124	**0.0886**	0.2054	**0.1680**
	0.20	0.1825	**0.1446**	0.1301	**0.0980**	0.2333	**0.1846**
	0.25	0.2002	**0.1531**	0.1458	**0.1062**	0.2580	**0.1995**
	0.30	0.2136	**0.1586**	0.1597	**0.1132**	0.2792	**0.2111**

Tables 5 and 6 correspond to the results obtained on both scenarios either on setup 1 or setup 2, using different standard deviation values for mutation. On scenario 1, the lowest MBF is obtained with a standard deviation of 0.20 for all subsets, on either setup. On scenario 2 the lowest MBF is achieved with a standard deviation of 0.01 on all subsets belonging to either setup.

The results obtained on scenario 1 seem to back up the assumptions from our previous analysis. A larger standard deviation will allow mutation to have a steeper effect on gene values and therefore a faster genetic drift through the search space, which seems relevant to finding better candidate solutions. However, the results show that above a certain bound, performance drops. This may be due to large standard deviations clashing with mutation constraints, which don't allow genes to stand out of their predefined interval. Such events may disrupt the positive effects of mutation and therefore cause performance to drop. A question remains if, even though larger

Table 7: Results obtained on both scenarios during the (source,sink) disturb study for setup 1. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.

Scenario	Std. Dev.	Subset 1		Subset 2		Subset 3	
		MAF	MBF	MAF	MBF	MAF	MBF
1	0.00	0.2048	0.1342	0.1724	0.1209	0.2537	0.1871
	0.05	0.1852	0.1579	0.1516	0.1328	0.2580	0.2278
	0.10	0.2149	0.1892	0.1615	0.1431	0.2737	0.2439
	0.15	0.2499	0.2255	0.1734	0.1554	0.2958	0.2664
	0.20	0.2845	0.2609	0.1853	0.1679	0.3296	0.3006
	0.25	0.3240	0.3007	0.2004	0.1837	0.3612	0.3325
	0.30	0.3640	0.3408	0.2198	0.2032	0.3935	0.3651
2	0.00	0.0538	**0.0523**	0.0476	**0.0466**	0.0823	**0.0806**
	0.05	0.0871	**0.0649**	0.0763	**0.0517**	0.1229	**0.0927**
	0.10	0.1291	**0.1065**	0.0813	**0.0642**	0.1416	**0.1132**
	0.15	0.1511	**0.1339**	0.0962	**0.0804**	0.1667	**0.1397**
	0.20	0.1849	**0.1687**	0.1127	**0.0977**	0.1948	**0.1692**
	0.25	0.2167	**0.2010**	0.1311	**0.1168**	0.2258	**0.2007**
	0.30	0.2521	**0.2367**	0.1514	**0.1377**	0.2549	**0.2302**

standard deviation values are disruptive, better results could be achieved by increasing even further the mutation probability while maintaining the standard deviation at an appropriate level.

The behaviour shown on scenario 2 also backs up the discussion regarding mutation probabilities. Not only does the algorithm seem to benefit from a smaller number of mutations on this scenario, Tables 5 and 6 show that smaller standard deviation values bring performance gains over larger values. As the algorithm only has to evolve disturbances added to estimated values, we expect these values to remain close to 0. Considering the initialization mechanism applied, using a small standard deviation for mutation will allow the search to remain focused on a small part of the search space, which given the characteristics of the scenario and the fact that individuals represent disturbance values, seems beneficial. Still, the tables show that when the standard deviation falls to the lowest value tested, the MBF slightly increases, suggesting that there is a lower bound, below which the performance of the search process is affected.

Table 8: Results obtained on both scenarios during the (source,sink) disturb study for setup 2. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.

Scenario	Std. Dev.	Subset 1		Subset 2		Subset 3	
		MAF	MBF	MAF	MBF	MAF	MBF
1	0.00	0.2044	0.1411	0.1831	0.1337	0.2826	0.2224
	0.05	0.2094	0.1821	0.1666	0.1493	0.2864	0.2603
	0.10	0.2333	0.2111	0.1742	0.1579	0.3130	0.2872
	0.15	0.2809	0.2600	0.1891	0.1735	0.3283	0.3041
	0.20	0.3232	0.3031	0.2065	0.1918	0.3616	0.3383
	0.25	0.3667	0.3477	0.2259	0.2120	0.3961	0.3733
	0.30	0.4136	0.3948	0.2493	0.2351	0.4362	0.4136
2	0.00	0.0467	**0.0455**	0.0471	**0.0462**	0.1025	**0.1009**
	0.05	0.1117	**0.0874**	0.0773	**0.0559**	0.1442	**0.1164**
	0.10	0.1525	**0.1338**	0.0932	**0.0782**	0.1737	**0.1490**
	0.15	0.2066	**0.1902**	0.1140	**0.1007**	0.2096	**0.1871**
	0.20	0.2565	**0.2413**	0.1366	**0.1247**	0.2489	**0.2276**
	0.25	0.3032	**0.2889**	0.1607	**0.1497**	0.2882	**0.2679**
	0.30	0.3466	**0.3326**	0.1856	**0.1751**	0.3303	**0.3107**

Tables 7 and 8 show the results obtained on the two scenarios on each subset using different standard deviation values for the disturbance applied on the costs of the desired shortest paths. On both setups and consistently on all subsets, the performance of the algorithm decreases as the standard deviation rises.

As previously stated, increasing the standard deviation in this case will increase the chance that incompatible shortest path costs exist and therefore reduce the chances that an optimal solution can be found. The results obtained are therefore expected. As the costs are disrupted based on a higher standard deviation, the more difficult the problem will be, causing the algorithm to perform increasingly worse on either scenario.

Also, regarding scenario 2 where cost estimates are used, increasing the standard deviation will cause the arc costs of the desired shortest paths to distance themselves from initial values. In consequence, the larger the standard deviation is, the worst the initial estimates get and the larger

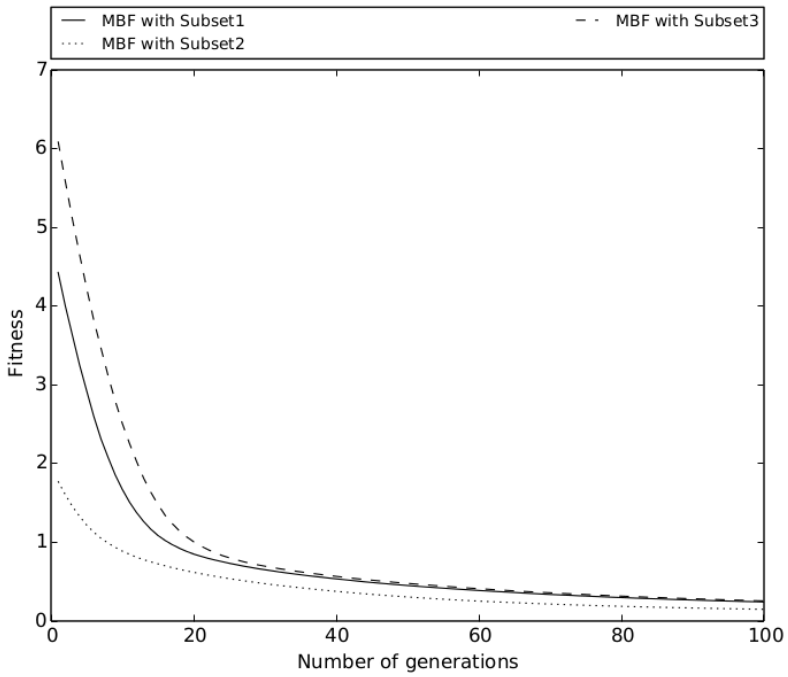

Figure 1: Mean Best Fitness (MBF) observed along the generations on scenario 1 with setup 1

~~the~~ effort from the algorithm will have to be. While the results obtained by the algorithm on both scenarios share characteristics, there are also some differences worth addressing. Firstly, we can observe that the results obtained on scenario 2 are consistently better than those obtained on scenario 1. More interestingly, we can also see that the results achieved on this scenario on the most difficult instances are comparable to those obtained on scenario 1 on the easiest instances.

These results show that the tasks proposed in scenarios 1 and 2 have quite different difficulty levels and that it may be often better to work with cost estimates, even if they are quite deteriorated, than starting from blind estimates. Scenario 1 appears to represent a more difficult task as the algorithm has to search the whole search space of arc costs. However, scenario 2 also seems challenging as the algorithm actually has a larger search space. However, the initial estimations allow it to use more appropriate initialization and mutation mechanisms, which give it a clear advantage. Finally, on both scenarios, but especially on scenario 2, the algorithm comes quite close to finding a valid solution on some instances of the experimental set. However, as seen by the relevant tables, the MBF is still higher than 0, showing that the problem is still quite challenging

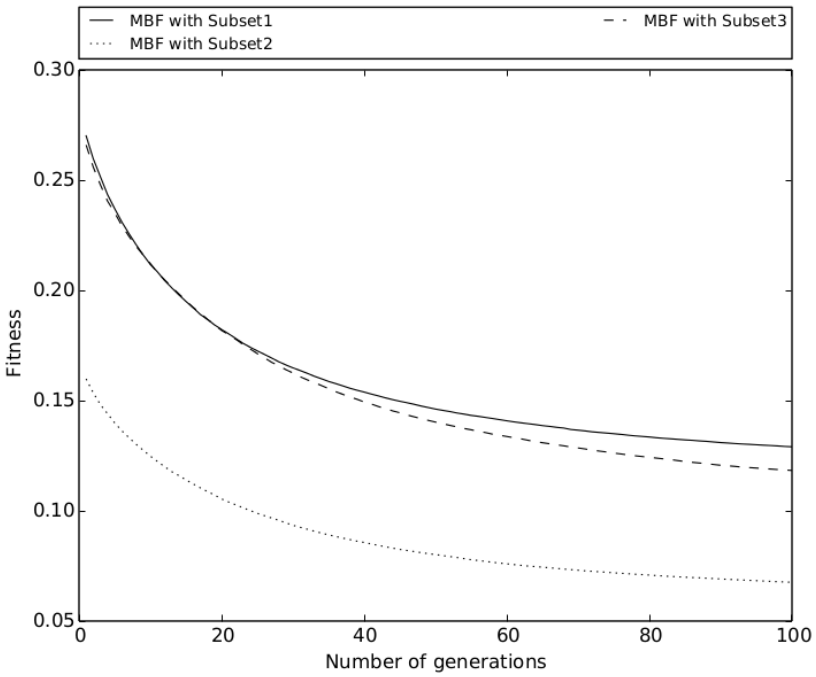

Figure 2: Mean Best Fitness (MBF) observed along the generations on scenario 2 with setup1

even when the desired shortest path costs are not disturbed. This is particularly relevant on the second scenario since an optimal solution in this case would be a vector of costs with value 0.

Apart from differences in behaviour obtained with different design options and scenarios, the overall results also show some differences on how the algorithm behaves when facing different subsets. These behaviours are observed in both scenarios over the entire parametric study and are worth addressing.

On both scenarios, best results were obtained on subset 2. The graphs from this subset consisted of 30 nodes and 100 arcs. Performance drops when the number of nodes is reduced to 15 (subset 1) but also when the number of arcs increases to 200 (subset 3). The subset where the algorithm achieves better results is therefore the one where the graphs are sparser, which may be an important characteristic. Having denser graphs means that there are more arcs connecting the nodes and therefore a higher number of alternative paths. While this may seem beneficial, especially considering that dealing with a high number of desired shortest paths means that there will be conflicts, it makes individuals more resilient to changes and therefore to adaptations. This may

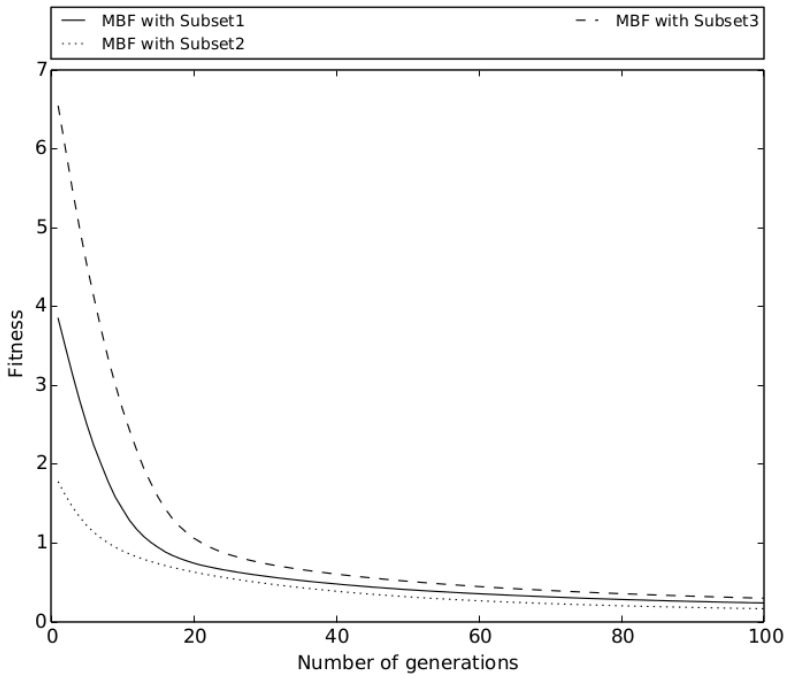

Figure 3: Mean Best Fitness (MBF) observed along the generations on scenario 1 with setup2.

compromise the evolution process enough to account for the differences in performance. Also, when generating $(source, sink)$ pairs, having a smaller number of nodes may end up creating a larger number of conflicts between desired shortest path arc costs and therefore increasing the difficulty of the problem. This may be responsible for the drop in performance of subset 1. Regarding subset 2, the drop in performance is probably linked to a higher search space resulting from doubling the number of arcs. The higher density will also make the individuals more resilient to adaptation.

Furthermore, setup 2 doubles the number of paths used in setup 1, causing performance to drop on all subsets. This behaviour is consistent with the above discussion, as increasing the number of paths will result on a higher number of desired shortest path cost conflicts and thus a more difficult challenge.

Figures 1 through 4 show the evolution of the mean best fitness along the generations on each setup. Figures 1 and 2 regard scenario 1 and figures 3 and 4 regard scenario 2. In both cases, the figures are in-line with our previous discussion. Notice that in scenario 1, due to initialization

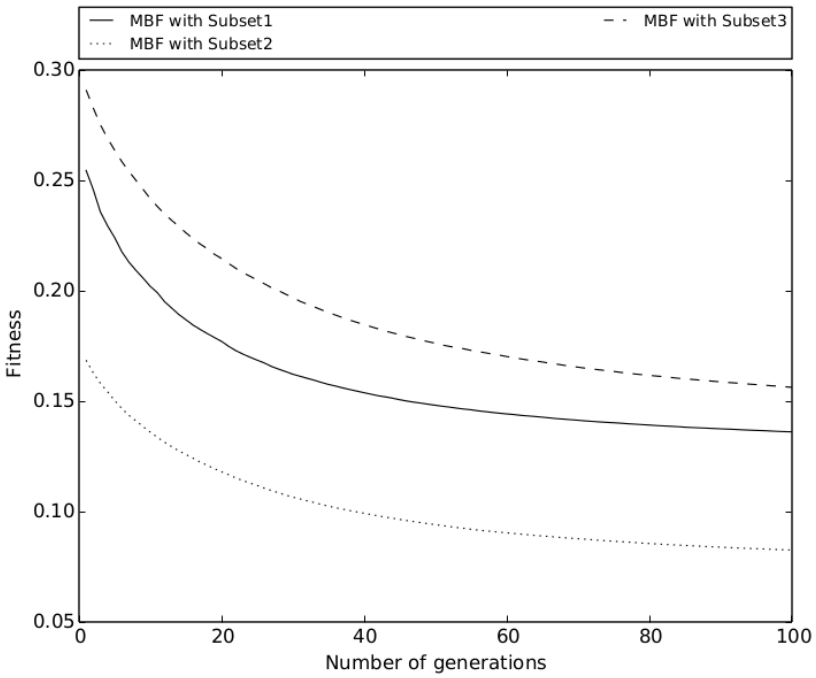

Figure 4: Mean Best Fitness (MBF) observed along the generations on scenario 2 with setup2.

using a uniform distribution, the algorithm starts from a much higher level in setup 3 but in only a few generations is able to close up on setup 1. The same thing does not happen with scenario 2, where the algorithm already starts with very close MBF values do to relying on a gaussian distribution for initialization. On both scenarios, the algorithm starts with a better MBF value on setup 2 when compared with the other two, probably due to the sparsity of the graph.

Tables 9 and 10 show the results obtained on both scenarios on setups 3 and 4 respectively. So far, the largest graphs tested had 30 nodes and 200 arcs. Setups 3 and 4 aim to test the algorithm in much larger graphs, having either 50 or 100 nodes, and 500 or 1000 arcs. What distinguishes setup 3 from setup 4 is the number of desired shortest paths. Setup 4 tests 3000 paths, which makes it a much more difficult challenge than setup 3 that tests 1000 paths.

Graphs this large represent a much tougher challenge to the algorithm, implying a much larger search space and computational effort. Due to this aspect, only 3 graphs were tested and 10 repetitions were run for each one. While a much larger number of experiments is needed to truly reach conclusions on the behaviour of the algorithm, the data obtained from these runs and

presented here is still worth addressing.

Table 9: *Results obtained on both scenarios using the best parameters found for setup 3. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.*

Scenario	Subset 1		Subset 2		Subset 3		Subset 4	
	MAF	MBF	MAF	MBF	MAF	MBF	MAF	MBF
1	0.4710	0.4449	0.3899	0.3702	0.5829	0.5534	0.4903	0.4622
2	0.2962	0.2716	0.1745	0.1604	0.4854	0.4507	0.2916	0.2714

Table 10: Results obtained on both scenarios using the best parameters found for setup 4. MBF stands for the mean best fitness and MAF for the mean average fitness obtained on all repetitions and graphs.

Scenario	Subset 1		Subset 2		Subset 3		Subset 4	
	MAF	MBF	MAF	MBF	MAF	MBF	MAF	MBF
1	0.4647	0.4424	0.3885	0.3696	0.5778	0.5531	0.5018	0.4773
2	0.3076	0.2860	0.1795	0.1672	0.5109	0.4805	0.2968	0.2795

The results obtained on both scenarios were considerably worse than those observed on setups 1 and 2. While this may be expected, it raises questions regarding the scalability of the algorithm to larger and larger graphs. Depending on its application and the accuracy required for the arc costs, the algorithm may or may not be appropriate. Still, the algorithm was able to perform better on scenario 2 than on scenario 1 on all setups and subsets, similarly to what had been observed on smaller graphs. This data backs up that having arc cost estimates may produce clear advantages.

There are other similarities between the behaviour observed with setups 3 and 4 and that previously observed in setups 1 and 2. In both setup 3 and 4, subset 3 has the same number of nodes as subset 1 but a larger number of arcs. The increased density results on a unfavorable increase in MBF which can also be observed between subsets 2 and 4. On the other hand, when the number of

arcs is maintained and the number of nodes increased, such as from subset 1 to subset 2 or from subset 3 to subset 4, the algorithm improves its performance.

The fact that this behaviour is consistent both in small or large graphs reinforces our assumptions. However, when comparing setup 3 with setup 4, which triples the number of desired shortest paths, we can see a slight drop in performance on some subsets but also the opposite. Interestingly, this behaviour had not been observed on smaller graphs, where setup 1 consistently performed better than setup 2. This suggests that such large graphs can accommodate a very large number of desired shortest path costs before we can observe the same effects we did between setups 1 and 2.

Conclusions and Future Work

Inverse Combinatorial Optimization has become a relevant and attractive research subject over the past decades. Several approaches have been proposed and studied to tackle both Inverse Solution Optimization Problems and Inverse Objective Value Optimization Problems. This study focuses on the ISPL problem, which is relevant to graph theory and may have several real world applications.

We propose an unprecedented GA to tackle the problem and two possible scenarios are presented. On the first scenario we assume that no information is accessible about the arc costs of the graph except the cost of the desired shortest paths. On the second scenario, we have access to estimates about the arc costs of the graph and only adjustments to be made to those estimates are evolved. A disturbance method is also introduced which by disturbing the arc costs belonging to the desired shortest paths, makes the problem tougher to solve as incompatible combinations may arise and as cost estimates become more and more deteriorated.

Both scenarios are tested using graphs with different characteristics. Two setups use graphs with a small size, up to 30 nodes and 200 graphs, and two setups use graphs of a larger size, up to 100 nodes and 1000 arcs. Different numbers of desired shortest paths costs are also tested, up to 200

on the smaller graphs and up to 3000 on the larger ones. A parametric study was conducted on the smaller graphs, where various parameter values were tested either regarding mutation probability, mutation standard deviation or shortest path disturbance's standard deviation. The results and behaviour of the algorithm on both scenarios are addressed and discussed. It is noted that the scenarios result on very distinct behaviours and the algorithm obtains its best performance with very different mutation parameters. While on the first scenario the algorithm benefits from higher mutation probabilities and standard deviation, on the second scenario it performs better with lower mutation probabilities and standard deviation. On the first scenario it is also observed that too high values of standard deviation can reduce the algorithms performance.

The effect of different values for the standard deviation of the shortest path disturbance method is also tested. The algorithm achieved a worst performance as the standard deviation gets larger on both scenarios, which was to be expected. The relation between performance of the algorithm and the characteristics of the graphs is also explored on both scenarios. Results show that on both scenarios, the algorithm perform better on sparser graphs as well as on setups with less desired shortest paths. However, on larger graphs, the differences are reduced and the results are competitive with each other.

Future work may include expanding the study to include graphs with different characteristics, e.g. graphs with one source node and many sink nodes. Also, a larger parametric study which includes tournament sizes or specific mutation operators may be relevant. Other representations as well as problem specific operators may also be tested. Finally, a GA approach could be applied not only to ISPL problems but also to ISPP or other inverse combinatorial optimization problems

References

Ahmed, S., & Guan, Y. (2004). The inverse optimal value problem. *Mathematical programming, 102*(1), 91-110.

Ahuja, R. K., & Orlin, J. B. (2000). A faster algorithm for the inverse spanning tree problem. *Journal of Algorithms, 34*(1), 177-193.

Ahuja, R. K., & Orlin, J. B. (2001). A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. *Operations Research, 49*(5), 784-789.

Ahuja, R. K., & Orlin, J. B. (2001). Inverse optimization. *Operations Research, 49*(5), 771-783.

Ahuja, R. K., & Orlin, J. B. (2002). Combinatorial algorithms for inverse network flow problems. *Networks, 40*(4), 181-187.

Bellman, R. (1956). *On a routing problem.* Tech. rep., DTIC Document.

Bertsekas, D. P. (1991). An auction algorithm for shortest paths. *SIAM Journal on Optimization, 1*(4), 425-447.

Burton, D., & Toint, P. L. (1992). On an instance of the inverse shortest paths problem. *Mathematical Programming, 53*(1-3), 45-61.

Burton, D., Pulleyblank, W., & Toint, P. L. (1997). The inverse shortest paths problem with upper bounds on shortest paths costs. Em *Network Optimization* (pp. 156-171). Springer.

Cai, M. C., Yang, X. G., & Zhang, J. Z. (1999). The complexity analysis of the inverse center location problem. *Journal of Global Optimization, 15*(2), 213-218.

Cai, M., & Yang, X. (1994). Inverse shortest path problems. *Operations research and its applications. First International Symposium, ISORA, 95*, pp. 19-22.

Cui, T., & Hochbaum, D. S. (2010). Complexity of some inverse shortest path lengths problems. *Networks, 56*(1), 20-29.

Day, J. (2002). Management of railroad impedances for shortest path-based routing. *Ph.D. thesis*.

Day, J., Nemhauser, G. L., & Sokol, J. S. (2002). Management of railroad impedances for shortest path-based routing. *Electronic Notes in Theoretical Computer Science, 66*(6), 53-65.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik, 1*(1), 269-271.

Fekete, S. P., Hochst{\"a}ttler, W., Kromberg, S., & Moll, C. (1999). The complexity of an inverse shortest path problem. *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, *49*, pp. 113-127.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM, 5*(6), 345.

Goldfarb, D., & Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, *27*(1), 1-33.

Gross, J. L., & Yellen, J. (2005). *Graph theory and its applications.* CRC press.

Hagberg, A., Swart, P., & S Chult, D. (2008). *Exploring network structure, dynamics, and function using NetworkX.* Tech. rep., Los Alamos National Laboratory (LANL).

Heuberger, C. (2004). Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, *8*(3), 329-361.

Hochbaum, D. S. (2003). Efficient algorithms for the inverse spanning-tree problem. *Operations Research, 51*(5), 785-797.

Hung, C.-H., & Director-Sokol, J. (2003). On the inverse shortest path length problem.

Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM), 24*(1), 1-13.

Liu, Z., & Zhang, J. (2003). On inverse problems of optimum perfect matching. *Journal of combinatorial optimization, 7*(3), 215-228.

Luke, S., Panait, L., Skolicki, Z., Bassett, J., Hubley, R., & Chircop, A. (2001). ECJ: a java-based evolutionary computation and genetic programming research system. *Dispon{\i}vel em http://www. cs. umd. edu/projetc/plus/ec/ecj/, {\'u}ltima visita em, 24*.

Mao-Cheng, C., & Li, Y. (1997). Inverse matroid intersection problem. *Mathematical Methods of Operations Research, 45*(2), 235-243.

Moore, E. F. (1959). *The shortest path through a maze.* Bell Telephone System.

Sokkalingam, P., Ahuja, R. K., & Orlin, J. B. (1999). Solving inverse spanning tree problems

through network flow techniques. *Operations Research, 47*(2), 291-298.

Xu, S., & Zhang, J. (1995). An inverse problem of the weighted shortest path problem. *Japan journal of industrial and applied mathematics, 12*(1), 47-59.

Yang, C., Zhang, J., & Ma, Z. (1997). Inverse maximum flow and minimum cut problems. *Optimization, 40*(2), 147-170.

Zhang, J., & Cai, M.-C. (1998). Inverse problem of minimum cuts. *Mathematical Methods of Operations Research, 47*(1), 51-58.

Zhang, J., Ma, Z., & Yang, C. (1995). A column generation method for inverse shortest path problems. *Zeitschrift f{\"u}r Operations Research, 41*(3), 347-358.

Zhang, J., Yang, X., & Cai, M. (2004). A network improvement problem under different norms. *Computational Optimization and Applications, 27*(3), 305-319.

Zhang, J., Yang, X., & Cai, M. (2004). Inapproximability and a polynomially solvable special case of a network improvement problem. *European Journal of Operational Research, 155*(1), 251-257.

Zhang, J., Yang, X., & Cai, M.-c. (1999). Reverse center location problem. Em *Algorithms and Computation* (pp. 279-294). Springer.

.