THESIS

GENERALIZED PARTITION CROSSOVER FOR THE TRAVELING SALESMAN PROBLEM

Submitted by

Douglas R. Hains

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2011

Master's Committee:

Department Chair: Darrell Whitley

Advisor: Darrell Whitley

Adele Howe

Jennifer Mueller

ABSTRACT OF THESIS

GENERALIZED PARTITION CROSSOVER FOR THE TRAVELING SALESMAN PROBLEM

The Traveling Salesman Problem (TSP) is a well-studied combinatorial optimization problem with a wide spectrum of applications and theoretical value. We have designed a new recombination operator known as Generalized Partition Crossover (GPX) for the TSP. GPX is unique among other recombination operators for the TSP in that recombining two local optima produces new local optima with a high probability. Thus the operator can 'tunnel' between local optima without the need for intermediary solutions. The operator is respectful, meaning that any edges common between the two parent solutions are present in the offspring, and transmits alleles, meaning that offspring are comprised only of edges found in the parent solutions. We design a hybrid genetic algorithm, which uses local search in addition to recombination and selection, specifically for GPX. We show that this algorithm outperforms Chained Lin-Kernighan, a state-of-the-art approximation algorithm for the TSP. We next analyze these algorithms to determine why the algorithms are not capable of consistently finding a globally optimal solution. Our results reveal a search space structure which we call 'funnels' because they are analogous to the funnels found in continuous optimization. Funnels are clusters of tours in the search space that are separated from one another by a non-trivial distance. We find that funnels can trap Chained Lin-Kernighan, preventing the search from finding an optimal solution. Our data indicate that, under certain conditions, GPX can tunnel between funnels, explaining the higher frequency of optimal solutions produced by our hybrid genetic algorithm using GPX.

TABLE OF CONTENTS

Chapter 1

Introduction

The Traveling Salesman Problem (TSP) is a well-studied combinatorial optimization problem with many applications and much theoretical value [App06]. The problem is stated as follows: given a set of n cities with a cost associated with each pair of cities, find a tour that visits each city only once and returns to the originating city with the lowest total cost.

The TSP is simple to state yet has been the subject of extensive research in the past one hundred years. As a result, there is a wide variety of algorithms that have been developed to solve the TSP (e.g., [JM97, LLKS85, App06]). These algorithms can be split into two broad categories: exact algorithms and approximation algorithms.

Exact algorithms produce a provably optimal solution [ABCC03, ABCC98, App06], but given that the TSP is NP-hard [GJ79], no exact algorithms are known which run in polynomial time for all problem instances. This can sometimes be an issue depending on the application for which the TSP is being solved. For example, using Concorde [ABCC03], a state-of-the-art TSP exact algorithm, to solve a 532 city problem takes roughly one minute on modern hardware. A 1,817 city problem on the same hardware takes over 8 days. One important application of the TSP is Very Large Scale Integration (VLSI) [App06], a process of creating integrated circuits. Problems with over 10,000 cities are common in VLSI[1]. By the time an exact solver could produce a solution to such a problem, the technology being designed by the VLSI would be obsolete.

Approximation algorithms run much faster than exact algorithms but are not guaranteed to produce the optimal solution. Approximation algorithms may be broken down further into two categories: tour construction and tour improvement algorithms. Tour construction algorithms begin with only a description of the particular TSP instance to be solved and incrementally construct a valid tour. Tour construction algorithms are quite fast but do not produce high quality solutions on their own; the constructed tour must be further improved by a tour improvement algorithm if a tour with a cost close to the global optimum is required.

The vast majority of tour improvement algorithms use some form of a local search heuristic at their cores. Although the solutions produced by tour improvement algorithms are not optimal,

[1] See http://www.tsp.gatech.edu/vlsi/

the best approximation algorithms typically produce solutions with costs within one percent or less of the optimal solution. The best performing class of approximation algorithms for the TSP is iterative local search [Hel00, ACR03, JM97].

Iterative local search heuristics traditionally work by improving only a single tour. Other classes of approximation algorithms work on an entire collection, or population, of solutions. One such class of approximation algorithms is the genetic algorithm (GA) [Whi94]. Although strict definitions of a GA exist, such as the original model proposed by Holland [Hol92] or the canonical GA of Vose [Vos99], in this work a GA is defined to be any algorithm which iteratively improves upon a population of solutions using *recombination* and *selection*. In this sense, many GAs have been developed for the TSP with much focus on the recombination operator [NK97, Nag06, TM98, Bra91, MGSK88]

Historically, GAs have not been as successful as local search based techniques for the TSP. Local search has been incorporated into GAs to form a Hybrid GA [MF97, WRE$^+$98]. For the TSP, Hybrid GAs typically outperform GAs that do not use local search, but are still not competitive with iterative local search algorithms. The choice of recombination operator is of particular importance when designing a hybrid GA. The operator cannot be too disruptive and undo the gains found by the local search heuristic but it must also introduce sufficient change to prevent the search from becoming stuck in local optima. In this thesis we present our recombination operator, called Generalized Partition Crossover (GPX) [WHH09, WHH] and a study of its performance in a hybrid GA alongside Chained Lin-Kernighan (Chained-LK) [ACR03], one of the best performing iterative local search algorithms.

The solutions produced by GPX are guaranteed to contain any links between cities that are common in the input tours. GPX will also not introduce any new links that are not found in the input tours. This helps to maintain subpaths found by the local search heuristic while also providing sufficient disruption to prevent the search from becoming stuck. GPX also has a unique ability we refer to as *tunneling*. Given two input tours that are locally optimal, there is a very high probability that the output tours will also be locally optimal. In this way, GPX is able to tunnel directly to new local optima.

To take advantage of the unique properties of GPX, we designed a hybrid GA (GPX-GA). We isolated the various design decisions and made these decisions in response to the specific properties of GPX. Of these decisions, one of the most important is the local search heuristic. In this study, we chose the Lin-Kernighan local search (LK-search) [LK73a].

Lin-Kernighan search is perhaps the most powerful local search for the TSP [JM97]. A number of improvements to LK-search have been proposed since its introduction in [LK73a]. The majority of these have been incorporated into the implementation used in Chained-LK [App06]. We use this same implementation of LK-search in GPX-GA for two reasons. One, it is highly efficient, and two, it gives us a straightforward way to compare GPX-GA and Chained-LK. This implementation of LK-search has a computational complexity of $O(n^2)$. The other components of each algorithm have linear time complexity, making LK-search the dominant factor in the running time of either algorithm. When given the same number of LK-search calls, we show that GPX-GA outperforms Chained-LK in terms of the quality of solutions found.

We note that although GPX-GA is able to find better tours than Chained-LK with similar amounts of computation, neither algorithm can consistently find the global optimum. While this is not a surprise, given that both algorithms are approximation algorithms, we investigate what might cause the algorithms to fail to find the global optima.

Chained-LK uses Lin-Kernighan search to find a locally optimal tour and then uses an operator known as the *double bridge* move to perturb the local optimum. The double bridge move pseudo-randomly selects four links between cities, breaks these links and replaces them with four new links chosen so that a valid tour is formed (See Figure 3.1 in Chapter 3).

We find that iterative local search frameworks using the double bridge move can become 'stuck' in search space structures we term funnels. Funnels are structures in the search space that form a type of gradient, guiding the search to their bottoms. The depiction in Figure 1.1(a) shows two funnels. These structures are similar to the funnels found in continuous optimization [DMW99] in that once an approximation algorithm finds the bottom of one funnel, it is very difficult to escape. We show that if Chained-LK becomes stuck in a funnel that does not contain the globally optimal solution, it is extremely unlikely that the global optimum will be found without restarting the search or employing some other mechanism to escape the funnel.

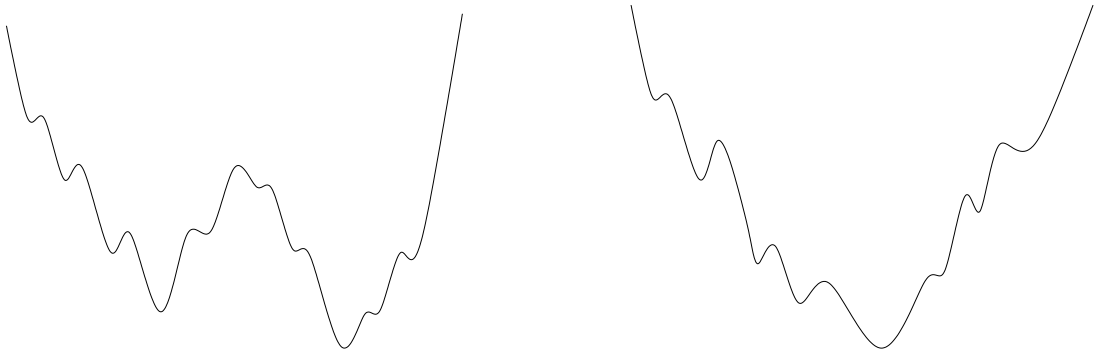


Figure 1.1: Two different search space structures: (a.) A depiction of a search space with two funnels. Smaller basins in each funnel can trap local search heuristics but can be escaped by meta-heuristics such as iterated local search. (b.) A depiction of the Big Valley hypothesis. The search space consists of a single central valley forming an overall gradient to the optimal solution.

We hypothesize that funnels are formed due to the break down in the Big Valley hypothesis [BKM94a, KT85] close to the globally optimal solution. The Big Valley hypothesis states that there is one large central valley where tours are correlated between evaluation and distance (See Figure 1.1(b)). This hypothesis implies that a gradient exists leading directly to the global optimal solution; a solution closer to the global optimum in terms of evaluation should also be closer in distance.

We find the central valley splinters into distinct funnels under Chained-LK when the evaluation values approach that of the global optimum, counter to the Big Valley hypothesis described in previous literature [BKM94a, KT85]. Non-optimal funnels contain tours within $.01\%$ of the global optimal evaluation value but can be more than a quarter of the maximum distance in the search space from the globally optimal tour, much further than would be expected under the Big Valley hypothesis. We find that the distance between funnels is too large for Chained-LK to escape.

Although Chained-LK cannot escape funnels, we show that GPX is capable of recombining tours from different funnels and producing tours located in new funnels. As the local search

heuristics used in Chained-LK and GPX-GA are identical, the GPX recombination operator can be thought of as an alternative perturbation operator to the double bridge move used in Chained-LK. Because GPX can escape funnels and the double bridge move cannot, this can in part explain the performance difference between the two algorithms. Indeed, the same analysis applied to Chained-LK to find the funnel structure is applied to tours produced by GPX-GA and we find the funnel structure is smoothed out, enabling GPX-GA to "tunnel" between funnels, preventing the search from getting stuck in a single funnel as it does when using Chained-LK. This is again analogous to the funnels of continuous optimization where a common approach to escaping funnels is to transform the energy landscape so the funnels no longer exist [DW98, PP95].

This still leaves open the question of why GPX-GA sometimes fails to find the global optimal. We find under certain conditions GPX is unable to recombine tours to find the global optimum mainly as a result of the way in which the recombination constructs new tours. Further analysis reveals in a large percentage of trials that all of the globally optimal edges are present in the population after a very small number of iterations of the hybrid GA. This suggests an even more powerful use of GPX-GA: quickly creating an edge pool which could be used to initialize an exact TSP algorithm, greatly reducing the $O(n!)$ search space.

1.1 Summary of Contributions

The contributions in this document are as follows:

1. The presentation and analysis of Generalized Partition Crossover (GPX) [WHH09, WHH], a recombination operator for the TSP. We design a hybrid GA, GPX-GA, to take advantage of the properties of GPX and show that GPX-GA is capable of outperforming Chained-LK, one of the best performing approximation algorithms known as of this writing.

2. An analysis of why the approximation algorithms fail to find the global optimum. This analysis yields two major results:

   (a) An analysis of iterated local search algorithms reveals a structure in the search space we call funnels. Funnels consist of tours that are correlated in terms of evaluation and distance. We have found that Chained-LK cannot typically escape from a funnel once the search has found one. The evaluation of the best tours found within the funnels of a particular instance of the TSP are within small values of one another, yet these tours are separated by large distances in the search space – a contradiction to the correlation between evaluation value and distance predicted by the Big Valley Hypothesis. We show that GPX is not as susceptible to funnels as Chained-LK, which explains in part its superior performance.

   (b) GPX fails to find the global optimum as a result of restrictions imposed on the construction of children tours in order to ensure a valid Hamiltonian Circuit is constructed. Under certain conditions, due to the properties of GPX, even if the globally optimal edges are present in the parent tours being recombined, it will be impossible for GPX to produce the global optimum. This analysis provides several directions for future work.

Chapter 2

Generalized Partition Crossover

A recombination operator takes two or more solutions and constructs new solutions based on the initial solutions. Because at least two solutions are required as input, recombination operators are used in population based search heuristics such as evolutionary strategies (ES) [BS02] and genetic algorithms (GA) [Whi94]. In this chapter we present two versions of our recombination operator for the TSP.

The first, partition crossover, uses the union of two solutions to construct a graph and looks for a specific type of partition in the constructed graph. If such a partition is found, two unique children are then constructed.

The second, generalized partition crossover, is a generalized version of partition crossover. We find that in most cases, the union graph constructed of two children contains multiple partitions of the specific type partition crossover is looking for. Generalized partition crossover is capable of finding and using all of these partitions in a single recombination with no increase to the computational complexity of the operator.

In the terms of evolutionary computing, the solutions input to a recombination operator are called the *parents* and, similarly, the solutions produced by recombination are known as the *children* or *offspring*. According to [RS95], a recombination operator is *respectful* if and only if the children contain all alleles that are common to both parents. In the case of the solution representation used for partition crossover, an allele is a single edge within a solution. An operator is said to *transmit alleles* if each and every edge in a child tour is present in at least one of the parents.

Both partition crossover and the generalized version are respectful and transmit alleles. This sets apart our operator from other recombination operators for the TSP, such as edge recombination [WSF89], edge assembly [Nag97], maximum preservative crossover [GS02] and distance preservative crossover [FM96], as these operators can introduce edges which are not found in the parent solutions.

2.1 Partition Crossover

Partition Crossover (PX) is a recombination operator that we developed for the TSP [WHH09] and shares some similarities to the partial transcription method of [MFMS99]. PX makes use of the fact that a tour t can be represented by a Hamiltonian circuit on a graph with n vertices where

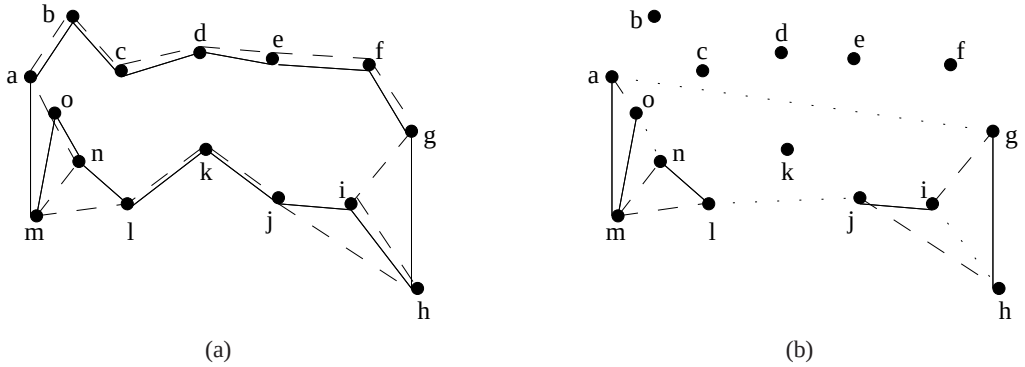


Figure 2.1: An example of (a) The union graph of two parent tours (one shown by solid lines, the other by dashed) and (b) The reduced graph $G*$ made by replacing common paths with surrogate edges (dotted lines).

there is a one to one mapping between the vertices in the graph and the cities in a given instance of the TSP. A cost is associated with each edge from vertex i to j, denoted $e(i, j)$, which is equal to the cost between cities i and j specified by the TSP instance. The total cost or evaluation of the tour, denoted $C(t)$, is the sum of the costs associated with each edge in the circuit. PX could potentially accept any representation of a Hamiltonian circuit. Our implementation assumes the tour is represented as a flat array, assuming an edge between the first and last elements.

Given two parent tours t_1 and t_2, PX first constructs a graph $G = (V, E)$ by taking the union of t_1 and t_2 so that V is the set of n vertices (i.e., cities) and E are the edges found in t_1 and t_2. An edge in E can be classified as either a *common* edge or an *uncommon* edge. An edge in E is a *common* edge if it is found in both parent tours; an edge is an *uncommon* edge if it is found in only one parent.

For example, Figure 2.1a depicts the union graph G of two tours. One parent tour is shown with solid lines the other with dashed lines. In this example the edges $e(g, h)$, $e(g, i)$, $e(h, j)$, $e(i, j)$, $e(l, m)$, $e(l, n)$, $e(n, m)$, $e(o, m)$, $e(o, a)$ and $e(a, m)$ are uncommon edges and the rest are common edges.

After G is constructed, PX constructs a reduced graph G^* from G. All paths of common edges are replaced by a *surrogate* edge connecting the end points of each path. Figure 2.1b depicts the reduced graph G^* created from G in Figure 2.1a. The common paths have been replaced by surrogate edges and are depicted by dotted lines.

Graph G is formed by the union of two Hamiltonian circuits and is thus connected, and by construction so is G^*. A *partition* on connected graph is created by removing or cutting edges until the graph is unconnected. The *cost of the partition* is the minimal number of edges that must be removed to create that partition. After G^* is created, PX will find a partition of cost 2 on G^* if one exists. In Figure 2.1b, a partition of cost 2 can be formed by removing the edges $e(a, g)$ and $e(l, j)$.

The following theorem states that if a partition of cost 2 exists, we can always form two children tours distinct from the parent tours and, additionally, that PX is respectful and transmits alleles.

The PX Theorem: *If a partition of size 2 exists on the union graph $G*$, then we can al-*

ways construct two new child tours that are distinct from the parent tours. PX is respectful and transmits alleles. [WHH09]

The proof in [WHH09] describes how we can take construct two distinct offspring from the union of two parent tours if a partition of cost 2 exists by alternating the paths taken through the components created by the partition. In the previous example using the parent tours in Figure 2.1:

```
a-o-n-m-l-k-j-h-i-g-f-e-d-c-b
```

and

```
a-m-o-n-l-k-j-i-h-g-f-e-d-c-b
```

PX would form the tours:

```
a-o-n-m-l-k-j-i-h-g-f-e-d-c-b
```

and

```
a-m-o-n-l-k-j-h-i-g-f-e-d-c-b
```

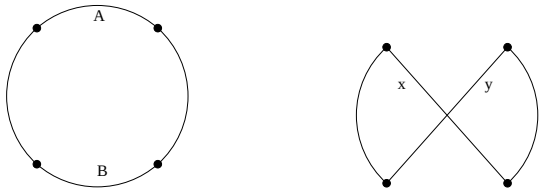
Of course, we cannot guarantee that a partition of cost 2 does exist in the union of two arbitrary Hamiltonian circuits. When a partition of cost 2 does exist, we say the recombination is *viable*. If the recombination is not viable, the operator produces offspring identical to the parents.

2.1.1 Tunneling Between Optima with PX

PX has another interesting ability we term "tunneling": Given two locally optimal parent tours, the children produced by PX are highly likely to be local optima as well. For tunneling to occur the parent tours need to be local optima, and therefore a local search heuristic is required.

For our initial tunneling experiments, we use a variation on the 2-opt algorithm [Cro58] known as steepest descent 2-opt. The algorithm works as follows. Given an initial tour, break a pair of edges and replace them with two different edges such that a valid tour is formed. The breaking and replacing of two edges is known as a *2-opt move* (See Figure 2.2). Record the new evaluation of the tour after the 2-opt move and restore the tour to its initial state. Repeat this process for every possible 2-opt move and take the move that yields the most improved tour. The entire process is repeated until no improving 2-opt move is found, at which point the tour is locally optimal.

We conduct the following two experiments to assess the ability of PX to tunnel between local optima in different parts of the search space. The first experiment recombines randomly generated local optima and the second recombines local optima that are close to the global optimum. We examine both sets of optima to determine if the tunneling property is affected by different areas of the search space. Additionally, we wish to examine if PX is capable of tunneling directly to the global optimum when recombining local optima known to be close to the global optimum.



(a) (b)

Figure 2.2: A 2-opt move. Edges A and B in (a) are broken and replaced with X and Y (b). Once A and B are broken, X and Y are the only edges that can be inserted to form a valid tour. The choice of the edge pair to be broken determines the 2-opt move.

Instance	rand500	att532	u574	nrw1379	rand1500	u1817
Viable recombination	14/45	43/45	33/45	40/45	37/45	45/45
Locally optimal offspring	96.4%	100%	98.5%	100%	97.3%	91.1%

Table 2.1: Number of recombinations in which a partition of cost 2 was found (viable recombination) and the percentage of offspring produced by PX that are locally optimal when recombining random local optima.

2.1.1.1 Recombining Random Local Optima

To determine how often PX produces local optima when recombining random local optima, we generated 10 unique local optima by applying steepest descent 2-opt to randomly generated tours. We recombined all possible unique pairs of the 10 local optima from each instance using PX. This resulted in $(10 \times 9)/2$ recombinations and 90 offspring. We counted the number of offspring from recombinations in which a partition of cost 2 was found and then reapplied steepest descent 2-opt to these offspring and counted the number that were not improved by the reapplication of steepest descent 2-opt.

We ran the experiment on six different TSP instances: rand500, att532, u574, nrw1379, rand1500 and u1817. att532, u574, nrw1379 and u1817 are from the TSPLIB[1]. The convention of TSPLIB is that the size of the problem, i.e. the number of cities, is represented by the numerical suffix of the name of the problem. We chose these instances as they represent two instances of different sizes from each the three general categories of TSP instances [JM97]: grid problems (u574 and u1817), pseudo-random or clustered problems (att532 and nrw1379) and random problems (rand500 and rand1500).

The results of this experiment are shown in Table 2.1 and tell us that PX tunnels to new local optima in at least 90% of recombinations if the parents are random local optima. We also see that the recombination was viable, i.e. a partition of cost 2 was present in the union graph, in the majority of attempts with the exception of rand500. The general trend is the larger the instance, the more often recombination is viable. To determine why this is the case, we counted the total

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Instance	rand500	att532	u574
Average number of partitions of cost 2 found	0.49 ± 0.79	2.07 ± 1.36	1.27 ± 1.08
Instance	nrw1379	rand1500	u1817
Average number of partitions of cost 2 found	1.91 ± 1.16	2.18 ± 1.77	4.00 ± 1.89

Table 2.2: Number of partitions of cost 2 found when recombining 10 random local optima.

Instance	rand500	att532	u574	nrw1379	rand1500	u1817
Viable recombinations	23/45	36/45	29/45	29/45	24/45	38/45
Locally optimal offspring	100%	100%	100%	100%	100%	100%
Globally optimal offspring	100%	77.8%	89.7%	37.9%	100%	100%

Table 2.3: Percentage of offspring produced by PX that are globally optimal.

number of partitions of cost 2 in the union graph of each recombination. The average number of partitions per union graph are shown in Table 2.2.

With the exception of att532, which appears to be somewhat anomalous, Table 2.2 shows a correlation in the number of partitions found and the size of the instance. We can conclude the larger the instance, the more partitions of cost 2 are found when recombining random local optima. PX will be able to produce offspring more often as the instance size grows and the offspring will not only be new solutions, but locally optimal solutions a high percentage of the time.

The properties of local optima found in the search space close to the global may be different than those when sampling random local optima. We wish to see if the viability of recombination and the tunneling property is maintained when recombining optima close to the global optimum. It may be possible for PX to tunnel directly to the global optimum if the search has reached a point close enough to the global optimum.

To produce local optima close to the global optima, we started from the globally optimal solution and modified it by making a random 2-opt move. After the random 2-opt move, the same 2-opt steepest descent heuristic used in the previous experiment was applied. If the resulting local optimum was different from the global optimum, we stopped. Otherwise, another random 2-opt move was made to the tour before steepest descent 2-opt was re-applied. The process was repeated until a local optimum distinct from the global optimum was found.

We found 10 unique local optima for each instance using this method. PX was then applied to each unique pair of these 10 local optima for a total of 45 recombinations. Of these recombinations, we count the number of recombinations in which a partition of cost 2 was found and, of the offspring produced by viable recombinations, how many were globally optimal. This experiment was conducted on the same instances as the previous experiment.

Table 2.3 shows that PX tunnels to new local optima in every case that the recombination was viable. This is most likely due to a higher concentration of local optima in the search space surrounding the global optimum then when sampling random local optima. Not only is PX able

Instance	rand500	att532	u574
Average number of partitions of cost 2 found	0.93 ± 0.96	1.22 ± 0.88	1.24 ± 1.05
Instance	nrw1379	rand1500	u1817
Average number of partitions of cost 2 found	1.00 ± 0.85	0.78 ± 0.82	1.56 ± 0.76

Table 2.4: Number of partitions of cost 2 found when recombining 10 local optima close to the global optimum.

to tunnel to new local optima, but it is sometimes able to tunnel directly to the global optimum. In the case of att532, PX was able to find an optimal solution different from the optimum tour provided by TSPLIB.

These results also show that, in general, the number of viable recombinations is reduced from those when recombining random local optima. We hypothesize this is due to greater similarity between tours close to the global optimum. As we are generating all the tours from the same solution (the global optimum), they will share more common edges than randomly generated local optima. Thus more edges will need to be removed to make the union graph disconnected causing partitions to have cost higher than 2.

Table 2.4 shows the counts of partitions of cost 2 when recombining local optima close to the global optimum. We see a decrease in the number of partitions of cost 2 from those reported in Table 2.2 with the exception of rand500. The rand500 instance is also the only instance in Table 2.3 in which the number of viable recombinations increased from that in Table 2.1. This shows a clear correlation in the number of partitions of cost 2 and the number of viable offspring. Furthermore, we can see that the average number of partitions of cost 2 decreases, with the exception of rand500, when recombining local optima close to the global optimum.

2.2 Generalized Partition Crossover

In our implementation of PX, only one partition of cost 2 is utilized in a single recombination even if there are multiple ways to partition G. For example, Figure 2.3a shows the union graph of two tours which have two partitions of cost 2, depicted by the heavy dark lines marked A and B. PX could construct two children using partition A in Figure 2.3a by taking the solid edges from the left of A and the dashed edges from the right, and a second child by taking the dashed edges from the left of A and the solid edges from the right (and inheriting all the common edges). Two different children would be constructed if PX used partition B in a similar manner.

In this example, although there are four possible children from a single application of PX, only two will actually be produced. To make use of all partitions between two parents, PX must be re-applied to the offspring. No edges are added because PX transmits alleles and is respectful, therefore any partitions not utilized in the parents will be found when recombining the offspring. If PX produces two children using partition A in Figure 2.3, partition B will be present if the children are recombined. In our implementation of PX if multiple partitions of cost 2 are present, PX will randomly choose one.

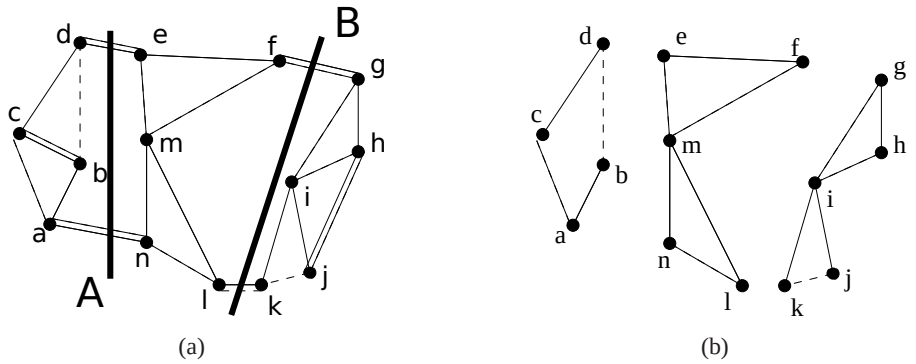


Figure 2.3: An example of (a) The graph G created from the union of two parent tours with two partitions of cost 2 shown by the heavy dark lines and (b) The graph G_u, constructed by deleting the common edges edges between the two parent tours from G

In general, if there are k partitions of cost 2 when recombining two tours there are $2^k - 2$ possible offspring. We find that as the instance size grows, the number of partitions found in a single union graph also grow. Our implementation of PX can only use one of these partitions in a single recombination, creating at most two distinct children out of a potential $2^k - 2$.

Generalized partition crossover (GPX) utilizes *all* partitions in a single recombination. Furthermore, using the greedy construction method described below to generate two children, one of the two is guaranteed to be the best solution out of all $2^k - 2$ possible offspring.

GPX works by creating a union graph $G = (V, E)$ of two parent tours just as PX. GPX then creates a subgraph of G, $G_u = (V, E_u)$, where V is the vertex set of G and E_u is the set of uncommon edges in E. Typically, G_u is made up of multiple disconnected subgraphs as shown in Figure 2.3b. In fact, if G_u is not disconnected, there is no way to partition G by only removing common edges.

We use breadth first search (BFS) on G_u to find each subgraph (i.e., connected components) of G_u. The BFS has a $O(n)$ computational cost, because the degree of any vertex is at most 4 and each vertex is processed only once. We will use the term *partition component* to refer to a connected component in G_u. A *feasible partition component* is a partition component which can be separated from the graph by a partition (or partitions) of cost 2.

For example, in Figure 2.3b, the left most partition component (consisting of vertices a, b, c, d) would be one side of a partition of cost 2 by cutting across A. This partition component is feasible. The partition component consisting of vertices g, h, i, j, k is also a part of a partition of cost 2 formed by cutting across B and is likewise considered feasible.

After identifying the components to the left of A and to the right of B, edges $e(d, e)$, $e(a, n)$, $e(f, g)$ and $e(l, k)$ would be marked as cut edges. GPX would then identify the partition component between A and B as feasible as all the common paths leading out from that component are also cut points in partitions of cost 2 on G_u.

Once no additional components are identified as feasible, the children tours are constructed. There are a number of possibilities for constructing children at this point. Our implementation begins by merging any remaining partition components which have not been identified as feasible into one large partition component. Children tours are then formed using the feasible

components and the large partition component. One child inherits the dashed edges from the large component and the other the solid edges. Both children then inherit the common edges and the shortest cost path through each of the feasible components.

As both children have the shortest cost path through each of the feasible components and only differ by the path through the largest component, one child must also have the shortest cost path through the largest component. This child will not only have a better evaluation then the other offspring produced in the recombination, but will have the best evaluation of all the $2^k - 2$ tours possible. Thus, GPX is capable of producing the best solution of all possible offspring *in a single recombination* with no increase to the running time of PX.

Although our implementation produces only two offspring, GPX can theoretically produce all possible offspring. We now present the GPX Theorem [WHH], showing that GPX is respectful and transmits alleles and can produce all possible $2^k - 2$ offspring in a single recombination.

The GPX Theorem:

Let graph G be constructed by taking the union of the vertices and edges found in two Hamiltonian Circuits for some instance of the TSP. If graph G can be separated into k partition components of cost 2, then GPX generates $2^k - 2$ distinct offspring; every recombination is both respectful and transmits alleles. [WHH]

The proof in [WHH] shows that GPX is respectful, transmits alleles and can generate all possible offspring. The number of possible offspring is a function of the number of feasible partition components (i.e., the number of partitions of cost 2) found in the union graph. While we have shown in theory that GPX can make use of all feasible partition components in a single recombination, we now wish to explore the benefits of generalization.

2.3 Benefits of Generalization

We will first look at the number of feasible partition components when recombining local optima in different areas of the search space. If only two partition components are used by GPX, GPX will perform no different than PX. We count the number of partitions of cost 2 found during recombination when recombining random local optima generated using three search heuristics: 2-opt [Cro58] and 3-opt [JM97]. 3-opt is similar to 2-opt but replaces three edges instead of two in a single move. As a result, 3-opt can find higher quality solutions than 2-opt [JM97]. Using both heuristics will allow us to sample different parts of the search space.

Two groups of 50 random tours were generated. Steepest descent 2-opt was applied to one group and steepest descent 3-opt was applied to the other. Both local search heuristics were applied until a local optimum was found. If a tour was duplicated within a group, a new random tour was generated and local search applied until each group consisted of 50 unique local optima.

GPX was applied to all unique pairs within each group. Table 2.5 shows the number of feasible partition components found in each recombination.

These results show that more partition components are present when recombining 3-opt local optima than 2-opt local optima because 3-opt local optima most share more common edges that can be used to partition the union graph. This makes sense, because the Big Valley Hypothesis [BKM94b] suggests an inverse correlation between the evaluation of local optima and the number of common edges shared with other local optima with similar evaluations. 3-opt finds local optima with lower evaluations than 2-opt, so by the Big Valley Hypothesis we would expect

Instance	rand500	att532	u574	nrw1379	rand1500	u1817
2-opt	2.6 ± 0.1	3.3 ± 0.2	3.1 ± 0.2	3.2 ± 0.2	3.7 ± 0.3	5.0 ± 0.3
3-opt	9.42 ± 0.4	10.5 ± 0.5	10.7 ± 0.5	11.3 ± 0.5	24.9 ± 0.2	26.2 ± 0.7

Table 2.5: Average number of *partition components* used by GPX in 50 recombinations of random local optima found by 2-opt and 3-opt.

more common edges between 3-opt local optima than 2-opt local optima.

In general, Table 2.5 tells us there are more than two partition components of cost 2 in most recombinations. But how does the use of multiple partition components impact the performance of the operator? We answer this question in part by looking at two different metrics: the tunneling ability of the operators and the improvement of the offspring over the parent tours when using PX and GPX.

Our goal of the following experiments is to examine the effect of using multiple partitions. We will again use 2-opt and 3-opt to generate the local optima. Table 2.5 shows that recombining 2-opt local optima generates a smaller number of partition components than using 3-opt local optima. This will allow us to contrast our results in terms of the number of partition components used.

We generated 10 random local optima using 2-opt and 10 random local optima using 3-opt by applying the local search algorithms to 10 random tours. Both algorithms are steepest descent.

We also generate 10 local optima near the global optimum by taking random walks from the global optimum as described previously. In the case of 2-opt, we use a random 2-opt move as a single step in the walk; in the case of 3-opt, we use a random 3-opt move.

For each set of local optima, we apply GPX and PX to each pair of optima resulting in 90 offspring for each set of 10. We measure the number of offspring that are improved over the parents and the number of offspring that are locally optimal (to measure the tunneling ability of the operator). When recombining parents near the global optimum, we also measure the number of offspring produced that are globally optimal.

2.3.1 Tunneling to Local Optima

Because GPX takes the best parts of each feasible partition component, we expect that the number of locally optimal offspring will increase when using GPX as compared to that of PX. The results of random local optima are in the top half of Table 2.6 and the local optima close to the globally optimum are in the bottom half.

While there is no clear trend in Table 2.6 that indicates an increase in the capability of GPX to tunnel to new local optima over that of PX, we can at the very least confirm that this property is not diminished in GPX. However we can see a trend indicating a decrease in the number of local optima produced when using 3-opt local optima from that produced by using 2-opt local optima.

We also hypothesize that GPX will be able to find the globally optimal solution more often than PX as a result of using multiple partitions. We calculated the percentage of successful recombinations that led directly to the global optimum and report these results in Table 2.7.

Table 2.7 indicates a clear ability for GPX to find the global more often than PX. This is due

(a) Percentage of local optima when recombining random locally optimal parents

	u574	u1817	rand500	rand1500	att532	nrw1379
2-opt (GPX)	97.4%	95.4%	94.4%	96.4%	100%	100%
2-opt (PX)	100%	91.1%	100%	97.1%	98.8%	97.5%
3-opt (GPX)	71.1%	48.9%	66.7%	50.0%	71.1%	77.8%
3-opt (PX)	71.1%	55.6%	64.4%	62.2%	64.4%	63.3%

(b) Percentage of local optima when recombining local optima near the global optimum

	u574	u1817	rand500	rand1500	att532	nrw1379
2-opt (GPX)	100%	100%	100%	100%	100%	100%
2-opt (PX)	100%	100%	100%	100%	100%	100%
3-opt (GPX)	98.7%	98.9%	91.7%	77.8%	100%	93.3%
3-opt (PX)	75.6%	76.1%	93.1%	57.1%	98.4%	59.8%

Table 2.6: The percentage of children tours that are locally optimal for different pairings of recombination and local search operators. Percentages are out of the total number of successful recombinations. (a) contains the results from recombining random local optima and (b) contains the results from recombining local optima near the global.

Instance	u574	u1817	rand500	rand1500	att532	nrw1379
2-opt (GPX)	89.6%	100%	100%	100%	77.8%	100%
2-opt (PX)	82.8%	100%	100%	100%	60.6%	100%
3-opt (GPX)	82.9%	95.4%	72.7%	71.4%	96.8%	83.3%
3-opt (PX)	41.4%	62.9%	41.4%	21.4%	58.0%	63.6%

Table 2.7: Percentage of recombinations that are globally optimal. Percentages are out of the total number of feasible recombinations, starting from random walk from global optimum for different pairings of recombination and local search operators.

to the fact that the generalized operator is capable of utilizing all partitions of cost 2 and constructing tours in a greedy manner. If more than one partition of cost 2 exists, PX must arbitrarily choose the correct partition to use to reconstruct the global. GPX will greedily construct the best child possible of all possible offspring. If it is possible to reconstruct the global optimum using the partitions in the union graph, GPX will find it. PX will only find the global optimum if it can be formed by using only a single partition and PX randomly chooses the correct partition.

We also see a lower percentage of times the global optimum is found when using GPX to recombine 3-opt local optima than when recombining 2-opt local optima. In Table 2.5, we saw a larger number of partitions of cost 2 when recombining random local optima generated by 3-opt. This is because 3-opt is a more powerful search than 2-opt and can find better tours.

When performing a random walk out from the global optimum, the walk must go further when using 3-opt because 3-opt is capable of finding the global optimum even when 2-opt cannot. Therefore, the 3-opt walks are finding on average tours that have worse evaluations than the 2-opt walks and share less edges with the global optimum. As a result, it is more difficult to find a combination of partition components that yields the globally optimal solution. It is impossible to tunnel directly to the global if all the globally optimal edges are not present in the partitions or if a common edge is not globally optimal.

This can also explain the greater difference between GPX and PX in the 3-opt results of Table 2.7 than the 2-opt results. Because the 3-opt local optima are further away from the global than the 2-opt local optima, it is more likely that a recombination must occur across multiple partitions to discover the global optimum. PX will be less likely to produce the global optimum as it can only use a single partition of cost 2. GPX is unaffected by the number of partitions of cost 2. If the globally optimal solution is present in the $2^k - 2$ possible offspring, our greedy implementation of GPX will find it.

2.3.2 Improvements over Parent Tours

We now look to see how the use of multiple partitions affects the improvement in evaluation values when recombining local optima. Using the offspring produced by recombining local optima in the previous experiment, we counted the number of offspring that were improved over the parents for each of the recombinations. To count as an improved offspring, a child must be improved over both parents. The percentage of improved children of the total offspring produced by successful recombinations are shown in Table 2.8.

In Table 2.8, we see that 50.0% is the highest percentage of children that PX can achieve. In PX, if one child is improved over both parents the other must be worse than both parents. Let $C(p_i)$ denote the cost of parent tour i and $C(c_i)$ denote the cost of child tour i, then

$$C(p_1) + C(p_2) = C(c_1) + C(c_2)$$

by construction of c_1 and c_2. Any decrease in evaluation of one child must result in an identical increase in evaluation of the other. Either both children are within the bounds of the parent evaluations and neither is improved over both parents or both are outside the bounds and one is improved and one is worse than both parents.

GPX is not restricted in this manner as it has multiple partitions from which it can construct offspring. As a result, it is capable of improving both offspring. Because GPX always con-

(a) Percentage of offspring improved over parent tours when recombining random locally optimal parents

Instance	u574	u1817	rand500	rand1500	att532	nrw1379
2-opt (GPX)	57.6%	72.2%	64.3%	75.7%	71.3%	70.0%
2-opt (PX)	50.0%	46.6%	50.0%	45.7%	48.8%	44.9%
3-opt (GPX)	100%	100%	98.9%	100%	98.9%	100%
3-opt (PX)	50.0%	48.9%	50.0%	50.0%	50.0%	50.0%

(b) Percentage of offspring improved over parent tours when recombining local optima near the global optimum

Instance	u574	u1817	rand500	rand1500	att532	nrw1379
2-opt (GPX)	87.9%	92.1%	91.3%	72.9%	66.7%	77.6%
2-opt (PX)	50.0%	50.0%	50.0%	45.7%	50.0%	50.0%
3-opt (GPX)	87.8%	98.8%	72.7%	71.4%	100%	100%
3-opt (PX)	50.0%	50.0%	50.0%	36.4%	50.0%	50.0%

Table 2.8: Percentage of children tours that have been improved over both parents for different pairings of recombination and local search operators. The results of recombining random optima are shown in (a). The results from recombining optima near the global are in (b).

structs the best offspring of the 2^k possible children it will find at least one improved offspring if possible.

Interestingly, although we saw a decrease in the number of offspring which were locally optimal in Table 2.6 when recombining 3-opt local optima, Table 2.8 shows a higher percent of improved evaluations when recombining 3-opt local optima. We attribute this to the fact that GPX has more degrees of freedom when recombining 3-opt local optima because of the greater number of partition components as indicated in Table 2.5.

GPX is able to improve the offspring more often, but we also examine if the amount of improvement in evaluation is greater than PX. Table 2.9 shows the average percent evaluation above the globally optimal solution for the random local optima from each instance before crossover is applied and after GPX and PX are applied.

We see that the offspring from GPX are generally more improved in all instances than those from PX. The use of multiple partitions not only increases the number of improved offspring, but also the evaluation of the improved offspring over those offspring produced from a single partition. We note the difference between PX and GPX is greater when using 3-opt local optima than 2-opt local optima. Again, this is attributed to the greater number of partition components available to the operator when recombining 3-opt local optima.

Partition crossover is a recombination operator that transmits alleles, is respectful and is capable of tunneling to new local optima. By using multiple partitions instead of a single partition, we retain all the desirable characteristics of the single partition version while increasing the number of improved offspring and the ability of the operator to tunnel to the global optimum. In the next chapter we will outline a hybrid genetic algorithm, GPX-GA, that we designed specifically for use with GPX and show that GPX-GA is capable of outperforming a state-of-the-art local search algorithm for the TSP.

Instance	2-opt	3-opt
rand500 - Before Recombination	10.20 ± 0.01	7.62 ± 0.01
rand500 - After PX	8.85 ± 0.01	7.19 ± 0.01
rand500 - After GPX	8.79 ± 0.01	6.91 ± 0.01
att532 - Before Recombination	9.04 ± 0.01	7.17 ± 0.01
att532 - After PX	8.08 ± 0.01	6.48 ± 0.01
att532 - After GPX	8.03 ± 0.01	6.20 ± 0.01
u574 - Before Recombination	9.93 ± 0.01	7.34 ± 0.01
u574 - After PX	9.22 ± 0.01	6.85 ± 0.01
u574 - After GPX	9.15 ± 0.01	6.53 ± 0.01
nrw1379 - Before Recombination	11.06 ± 0.01	7.86 ± 0.01
nrw1379 - After PX	10.58 ± 0.01	7.66 ± 0.01
nrw1379 - After GPX	10.52 ± 0.01	7.49 ± 0.01
rand1500 - Before Recombination	11.39 ± 0.01	8.23 ± 0.01
rand1500 - After PX	10.91 ± 0.01	7.94 ± 0.01
rand1500 - After GPX	10.87 ± 0.01	7.66 ± 0.01
u1817 - Before Recombination	15.36 ± 0.01	9.76 ± 0.01
u1817 - After PX	14.70 ± 0.01	9.18 ± 0.01
u1817 - After GPX	14.64 ± 0.01	8.60 ± 0.01

Table 2.9: The mean percentage and standard error above the optimal tour of the children tours produced by the successful recombinations for PX and EPX in each experiment using random locally optimal parents.

Chapter 3

GPX-GA: A Hybrid Genetic Algorithm using GPX for the TSP

3.1 Designing a Hybrid Genetic Algorithm for GPX

The general definition of a genetic algorithm (GA) used in this thesis is any search heuristic which uses recombination and selection to optimize a population of solutions. A hybrid genetic algorithm is a GA that, in addition to recombination and selection, also uses uses a local search heuristic. Given that GPX is capable of tunneling when recombining local optima, a hybrid genetic algorithm is a reasonable choice to leverage the unique properties of GPX as a recombination operator. We designed a hybrid GA (GPX-GA) using the following basic structure:

1. Initialize a population of t solutions.
2. Apply a local search heuristic to each of the t solutions.
3. Apply recombination to the population of t solutions, creating o offspring.
4. Select t tours from the o offspring to form the next generation.
5. Repeat from step 2 until some stopping criterion is met.

3.1.1 Initial Population

To create the initial population, we choose to create t random tours. Other tour construction algorithms, such as Quick-Boruvka [ACR03], Christofides [CG76] or nearest neighbor [JM97], produce better starting tours but as the next step is the application of local search, such gains are negligible as the local search algorithm used in our GPX-GA is much more powerful than even the best performing construction heuristics [JM97, App06].

3.1.2 Local Search

Currently, the most powerful local search heuristic as of this writing for the symmetric TSP is the Lin-Kernighan local search heuristic (LK-search) [LK73b]. It is not surprising that modified

versions of the original LK-search are used in the best performing iterative local search algorithms, such as Chained Lin-Kernighan (Chained LK) [ACR03] and Lin-Kernighan-Helsgaun (LKH) [Hel00]. We conducted pilot studies using 2-opt, 3-opt and LK-search in a hybrid GA with GPX. LK-search consistently produced better results when used in conjunction with GPX and is the natural choice for use as the local search heuristic.

LK-search performs a next descent backtracking search on an initial tour, exploring the neighborhood of tours found by making k-opt moves on the initial tour, as well as those tours that can be found by making $(k-1)$-opt moves, $(k-2)$-opt moves, etc. down to the basic 2-opt move. It explores this neighborhood by first breaking 2 edges and replacing them as in the 2-opt algorithm, but if the 2-opt move does not yield an improved tour, it retains the replaced edges and continues by breaking and replacing another edge to form a 3-opt move. This continues until an improved tour is found or k edges have been broken and replaced (i.e., a k-opt move has been performed). If an improved tour is found, the search is restarted from the improved tour, otherwise the search backtracks by undoing the last broken edge and breaking a different edge. When no improved tours are found within the k-opt neighborhood, the tour is a local optima and the search is complete.

The LK-search algorithm's ability to find improving tours and its efficiency have both been improved by a number of clever optimizations since its first description in [LK73b]. In this study, we use an efficient and effective implementation of the LK-search method that is used in Chained LK [ACR03] and is part of the Concorde software suite[1]. This particular implementation of LK-search is described in detail in [App06]. For the remainder of this thesis, LK-search refers to the particular implementation in version 03.12.19 of the Concorde software package unless otherwise noted.

Two important details of this implementation are the use of candidate lists and "don't-look" bits. Each city has associated with it a candidate list of l cities that are sorted by their likelihood of producing an improving move. One common way of sorting these lists is by edge cost; however, LK-search uses a Delauny graph to construct and sort the candidate lists (See [App06] for details). The candidate lists are used to decide which edge to break next when performing the search. By default, l is much smaller than $(n-1)$, thus not only are edges more likely to improve the tour chosen first, but the use of candidate lists also reduces the options that must be considered when backtracking.

LK-search also uses "don't-look" bits. "Don't-look" bits are used to prevent undoing an edge replacement that was performed previously in the same search. Even though a city may be in a candidate list, it will be overlooked if its "don't-look" bit is turned on. Because LK-search takes the first improving move found and the ordering of the initial cities involved in the first step of a variable k-opt move is random, it is possible (and highly likely) that the candidate lists will be consulted differently in multiple runs given the same initial tour.

Because of this stochasticity, a tour returned by LK-search is not a true local optimum in the sense that if a different candidate list is chosen on repeated applications, different search paths will be considered and the possibility of further improvement arises.

[1] http://www.tsp.gatech.edu/concorde.html

3.1.3 Recombination

Obviously GPX is the recombination operator used in GPX-GA, specifically the greedy implementation described in the previous chapter. We have shown that GPX transmits alleles and is respectful. This means that no new edges can be introduced into a tour via GPX alone. Furthermore, GPX tunnels between local optima, meaning a high percentage of the tours produced by GPX will also be locally optimal and will not be changed by LK-search. Without some mechanism to introduce new edges into the population, we find that GPX-GA converges prematurely before a low cost solution is found.

To increase the number of unique edges in the population and to prevent premature convergence, when a recombination is not viable we apply the double bridge move to the parents. The double bridge move selects four edges as random, breaks those edges and reconnects them as depicted in Figure 3.1.

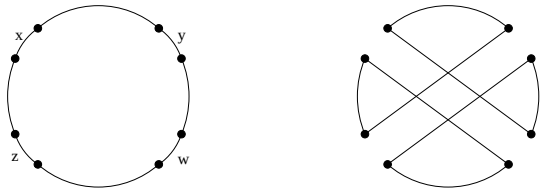


Figure 3.1: A double bridge move. Edges x, y, w and z in (a) are chosen randomly. They are then broken and replaced as shown in (b).

We use the implementation of random double bridge moves found in the Concorde software package and used in Chained-LK.

One further question that arises with recombination is what tours should be recombined. We first considered recombining all unique pairs of solutions in the current population. We found empirically that this method produced a number of redundant solutions and was not as effective as simply recombining the best tour of the population with the remaining $t-1$ solutions. Performing recombination in this manner did not significantly reduce the evaluation of the tours found after a fixed number of generations and eliminated a large number of recombinations in each generation.

3.1.4 Selection

In the greedy implementation of GPX used in GPX-GA, only two offspring are generated. One of the offspring is the greedy offspring: it takes the shortest path through each of the smaller feasible partition components and the shortest path through the largest component. The second offspring also inherits the shortest path in all of the partition components, except for the largest partition component; in this component the second offspring inherits the path *not* used by the first greedy offspring. Thus, t recombinations produces $2t$ offspring.

Several options exist for reducing the $2t$ offspring to form the next population of size t. We considered two options: 1) truncation selection, in which we always keep the t best solutions or 2) diversity selection, in which we try to preserve diversity by keeping offspring containing edges that are under-represented in the population.

Truncation selection is fairly straightforward and only involves the evaluations of each of the $2t$ tours produced. The t tours with lowest evaluation move forward to the next generation.

To maintain diversity, we developed a strategy called *diversity selection* that uses an edge weighting function d to quantify the diversity of edges contributed to the population by each tour. For tour s_i in the population,

$$d(s_i) = \sum_{e(j,k) \in s_i} \frac{1}{M(j,k)}$$

where $e(j,k)$ is an edge from city j to k and $M(j,k)$ is the number of times $e(j,k)$ appears in the population. We then retain tours from among the offspring with the highest summed edge diversity, $d(s_i)$.

The use of diversity selection means that the GA must be generational and that offspring replace parents, because parents typically have higher diversity than offspring.

To determine which selection method would be best at producing low cost tours and maintaining diversity, we ran the GA using both truncation and diversity selection for 100 generations with a population of 10 solutions (1,010 LK-search calls). At each generation we record the minimum tour evaluation in the population and the number of unique edges found in the entire population of solutions. Figure 3.2 shows the average results over 100 trials on the indicated instances. The instances are the same from the previous chapter and represent one large and one small instance from each of the three categories of TSP instances: grid (u574, u1817), clustered or psuedo-random (att532, nrw1379) and random (rand500, rand1500).

The left column shows the average minimum tour found and the right shows average number of unique edges in the population.

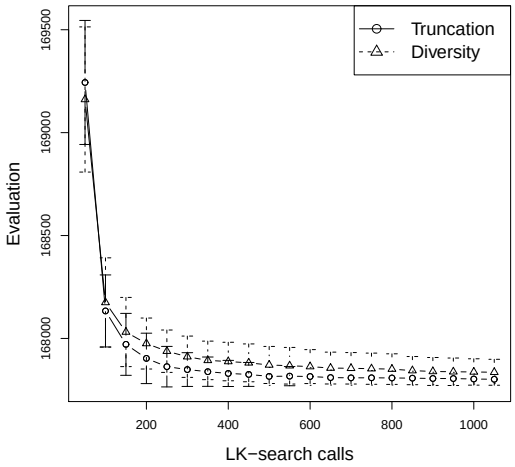
We see that the minimum evaluation found in both diversity and truncation selection are within the standard error (left hand figures of Figure 3.2). However, the right hand side of Figure 3.2 shows that diversity selection is able to maintain a much larger number of unique edges in the population. Because GPX is unable to introduce new edges into the population, we use diversity selection in the GA to maintain a larger number of unique edges without any significant loss in the minimum tour found.

3.1.5 Population Size

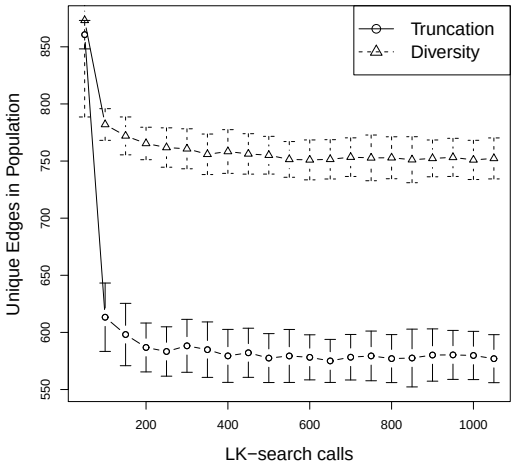
Figure 3.3 lists the pseudocode for GPX-GA, which incorporates the various algorithmic decisions described above. We now examine the question of finding the best value of t, or population size.

We conducted empirical experiments on the population size and determined that a population of size 10 does not perform significantly worse than populations of greater size and, in most cases, performs better. Figure 3.4 shows the results of running GPX-GA with various values of t.

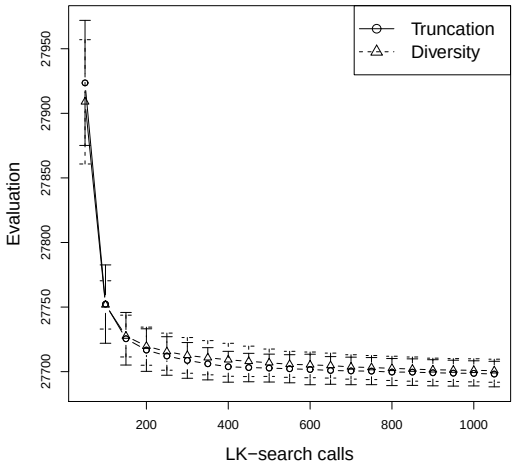
LK-search takes the most computational time in a single generation; for every increase in population, the number of LK-search calls per generation is also increased. Therefore, a population of size 20 has made 120 LK-search calls by generation 5 (including the 20 for initialization). For the same number of LK-search calls, a population of size 10 could be run for 21 generations. We chose a population of size 10 as the results indicate this value leads to the best performance and allows us to carry out more LK-search calls per tour.



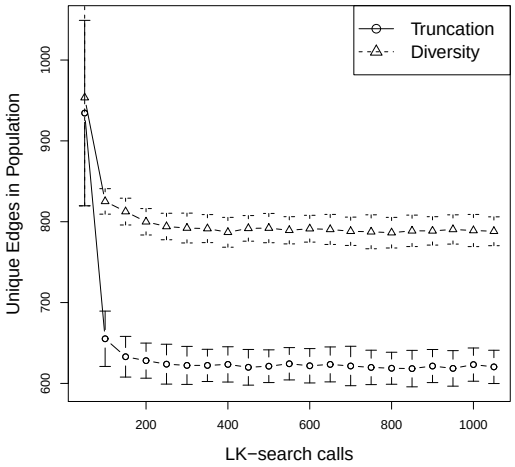
(a) rand500



(b) rand500

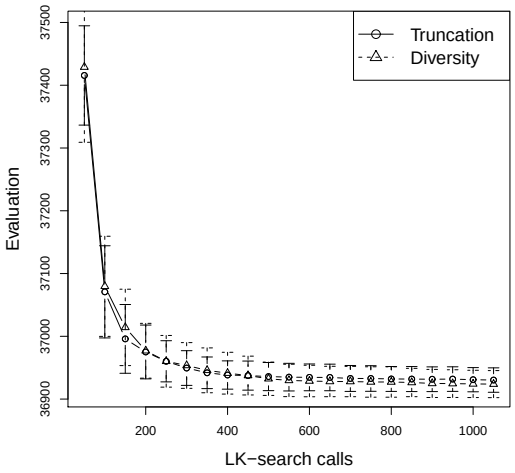


(c) att532

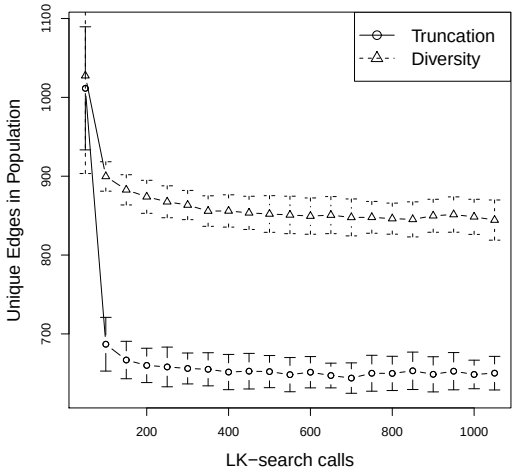


(d) att532

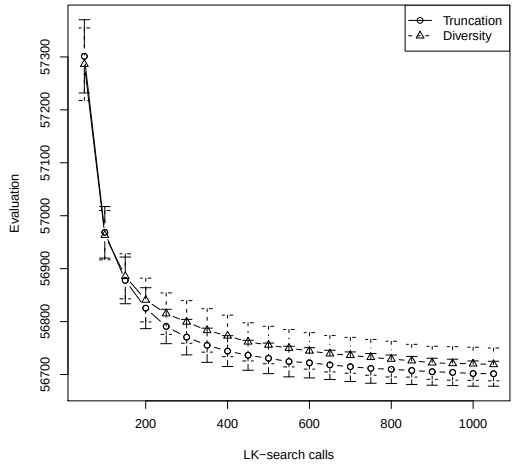
Figure 3.2: Comparing diversity and truncation selection on six instances. The average minimum tour found is shown on the left side shows no significant difference between selection methods. However, the right side shows a large increase in the number of unique edges in the population between the two methods.



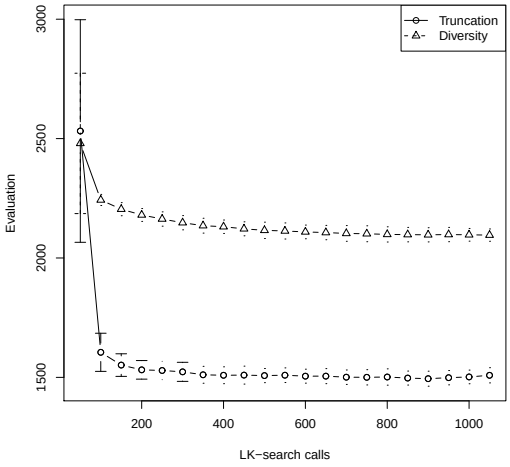
(e) u574



(f) u574

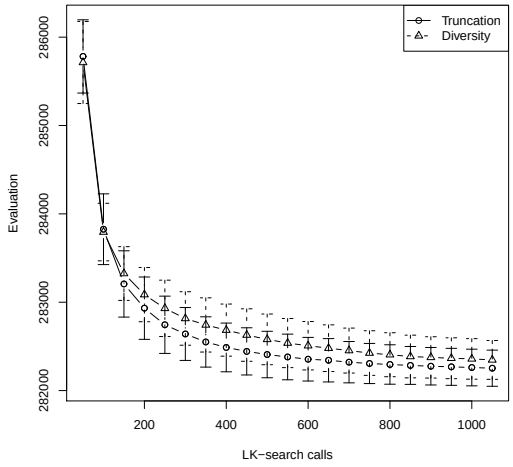


(g) nrw1379

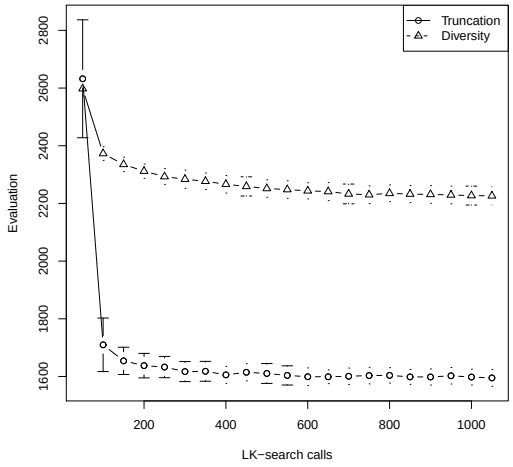


(h) nrw1379

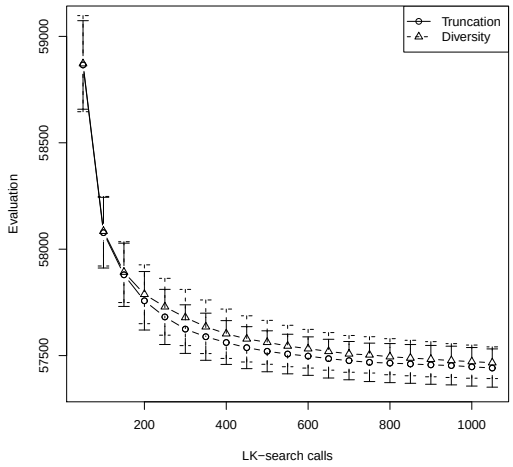
Figure 3.2: (cont.) Comparing diversity and truncation selection on six instances. The average minimum tour found is shown on the left side shows no significant difference between selection methods. However, the right side shows a large increase in the number of unique edges in the population between the two methods.



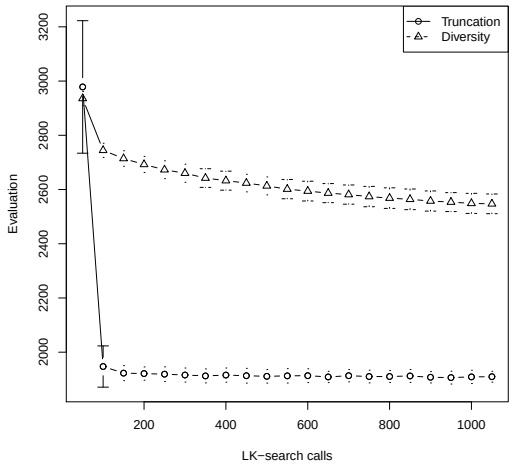
(i) rand1500



(j) rand1500



(k) u1817



(l) u1817

Figure 3.2: (cont.) Comparing diversity and truncation selection on six instances. The average minimum tour found is shown on the left side shows no significant difference between selection methods. However, the right side shows a large increase in the number of unique edges in the population between the two methods.

Let $P1$ be a randomly generated population of size t; Let $P2$ be a temporary child population of size t; For each member of $P1$: apply LK-search and evaluate;
1. Attempt to recombine the best tour of $P1$ with the remaining $t-1$ tours using GPX; this generates a set of up to $2t$ offspring. 2. If recombination was not feasible between the best tour and tour i, mutate parents using a double bridge move and place in population $P2$; 3. Place the best solution found so far in population $P2$; 4. From the set of offspring, select offspring to fill population $P2$; 5. For each member of population $P2$: apply LK-search and evaluate; 6. $P1 = P2$; If stopping condition not met, goto 1;

Figure 3.3: Algorithm for GPX-GA; the GA is generational, but elitist.

3.2 GPX-GA versus Chained-LK

To measure the performance of GPX-GA, we compare it to that of Chained-LK. Chained-LK is an iterative local search heuristic and is one of the best performing heuristics for the TSP as of this writing [JM97, ACR03]. A high level description of the iterative search framework used in Chained-LK is described in Figure 3.5. For a detailed description of Chained-LK see [App06].

To compare GPX to Chained-LK, we ran GPX-GA with a fixed number of available LK-search calls for the entire population. When these calls are depleted, the algorithm is terminated. Chained-LK is run with the same number of available LK-search calls *on a single tour*. Both algorithms use the same implementation of LK-search with the same default parameters.

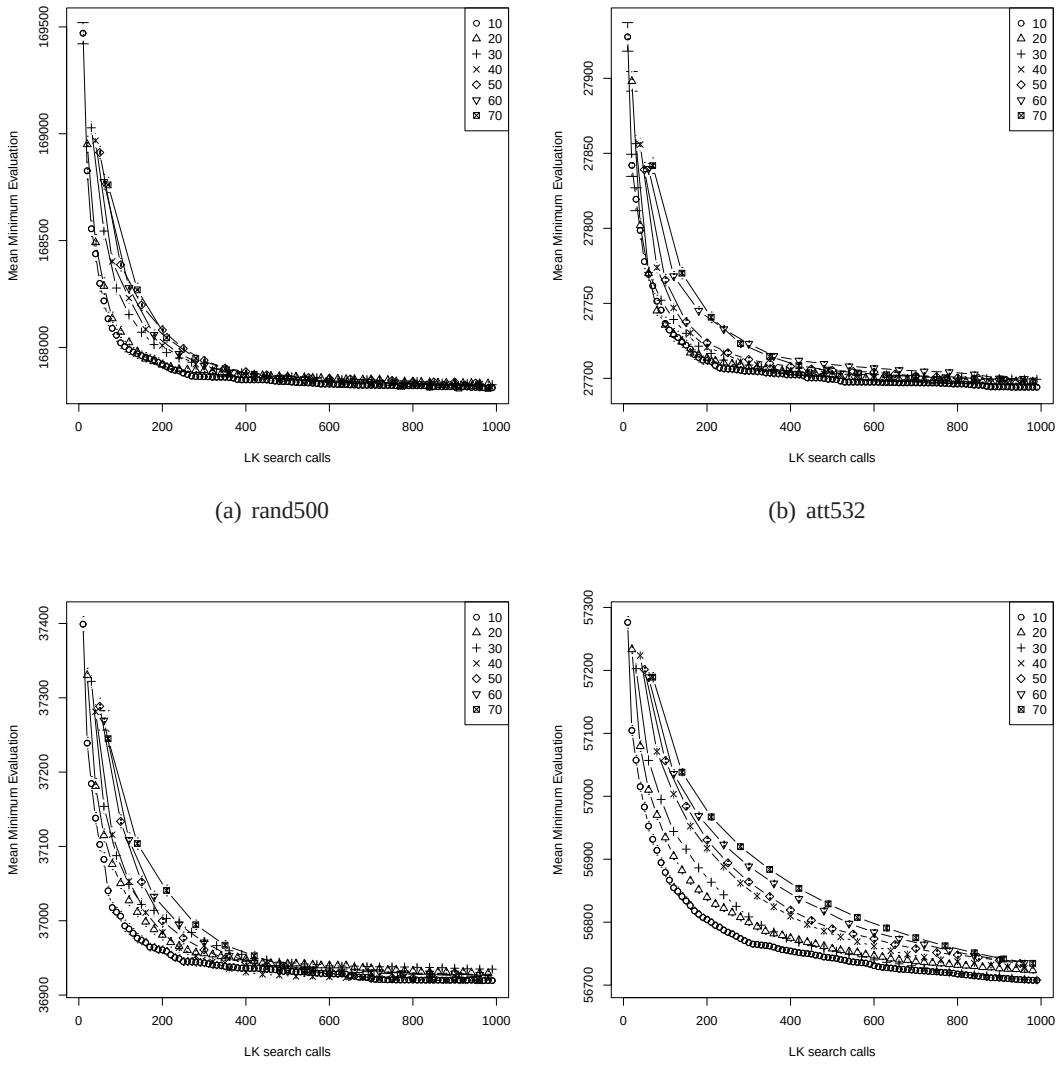
In the results presented here we ran GPX-GA with a population of 10 tours. Since the population size is 10, GPX-GA uses 10 applications of LK-search each generation; therefore, Chained-LK is allowed to do 10 double bridge moves and apply LK-search 10 times for every generation that GPX-GA is allowed to execute.

This means that each algorithm is allowed to call LK-search exactly the same number of times. The GPX-GA has the additional cost of recombination, but this cost is $O(n)$ with a small constant (of 4 or 5) and the computation is very small compared to one iteration of LK-search. Furthermore, applying LK-search after a double bridge move is more expensive that applying LK-search after recombination: the solutions are made poorer by the double bridge move, and improved by recombination.

We then compare the minimum tour found using GPX-GA against the minimum tour found using Chained-LK. Table 3.1 lists the average percentage of the cost of the minimum tour found compared to the cost of the global optimum for each problem instance. GPX-GA was allowed to run for 50 generations in these experiments.

After 510 calls each to LK-search (50 generations plus an initialization step), GPX-GA yields better results on all of the problems except nrw1379. This is remarkable because GPX-GA must optimize 10 solutions and the best solution must be optimized 10 times faster than Chained-LK to obtain a better result.

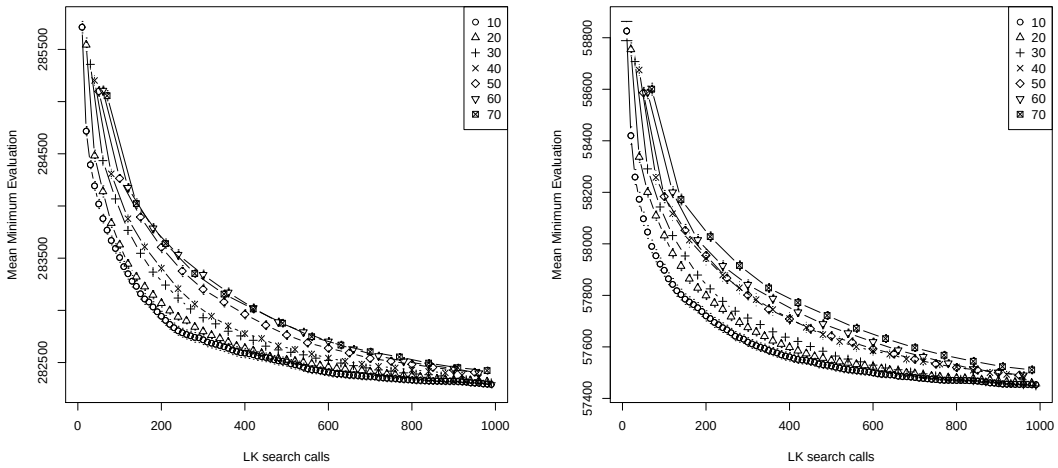
If each algorithm is run longer, the performance of GPX-GA is increasingly better than Chained LK. Table 3.2 shows how many times (out of 50 attempts) that each method finds the



(a) rand500 (b) att532

(c) u574 (d) nrw1379

Figure 3.4: The average minimum evaluation value found for various population sizes averaged over 100 trials.



(e) rand1500 (f) u1817

Figure 3.4: (cont.) The average minimum evaluation value found for various population sizes averaged over 100 trials.

Generation $\longrightarrow$		5	10	20	50
Instance	Algorithm	60 LK calls	110 LK calls	210 LK calls	510 LK calls
rand500	GPX-GA	0.29 ± 0.01	0.17 ± 0	0.1 ± 0	0.05 ± 0
	Chained-LK	0.30 ± 0.01	0.19 ± 0.01	0.13 ± 0	0.09 ± 0
att532	GPX-GA	0.29 ± 0	0.18 ± 0	0.12 ± 0	0.07 ± 0
	Chained-LK	0.30 ± 0.01	0.21 ± 0.01	0.13 ± 0	0.08 ± 0
u574	GPX-GA	0.49 ± 0	0.33 ± 0	0.19 ± 0	0.15 ± 0
	Chained-LK	0.54 ± 0.01	0.34 ± 0	0.19 ± 0	0.15 ± 0
nrw1379	GPX-GA	0.63 ± 0	0.48 ± 0	0.34 ± 0	0.23 ± 0
	Chained-LK	0.62 ± 0.01	0.46 ± 0.01	0.32 ± 0	0.19 ± 0
rand1500	GPX-GA	0.71 ± 0.01	0.52 ± 0.01	0.36 ± 0	0.22 ± 0
	Chained-LK	0.73 ± 0.01	0.54 ± 0.01	0.39 ± 0.01	0.25 ± 0
u1817	GPX-GA	1.61 ± 0.01	1.26 ± 0.01	0.95 ± 0.01	0.63 ± 0.01
	Chained-LK	2.08 ± 0.02	1.61 ± 0.02	1.19 ± 0.01	0.83 ± 0.01

Table 3.1: Average percentage of the cost of the minimum tour found above the globally optimal cost averaged over 500 experiments using Chained LK and GPX-GA.

Let t be a valid tour;
1. Let $s = t$ 2. Apply double bridge move to s 3. Apply LK-search to s 4. If $C(s) < C(t)$, let $t = s$ 5. Goto 1 until stopping criterion is reached.

Figure 3.5: High level overview of the iterated local search framework used by Chained-LK. The default stopping criterion is to run the main loop for n iterations where n is the number of cities in the TSP instance being searched.

	rand500	att532	u574	nrw1379	rand1500	u1817
Hybrid GA	50/50	26/50	43/50	1/50	12/50	1/50
Chained-LK	38/50	16/50	32/50	1/50	2/50	0/50

Table 3.2: This data reports the number of times the global optimum is found by each algorithm after 1010 calls to LK-search. The results are report over 50 experiments.

global optimum after 1010 calls to LK-search (which is 100 generations for GPX-GA).

We have shown that when given similar computational resources, GPX-GA is capable of finding better tours faster than Chained-LK and finding the global optima more often. In this chapter, we have concentrated on the performance of these algorithms and how often they find the global optimum. In the next chapter, we will explore what happens when these algorithms fail to find the global optimum.

Chapter 4

Funnels in the Search Space of the Travelling Salesman Problem

The neighborhood of the TSP has been described as having a big-valley structure [BKM94a] in that the evaluation of a tour is positively correlated to the distance of the tour from the global optimal and to other locally optimal tours (see Figure 4.1). In [BKM94a], many local optima were generated by Lin Kernighan search (and other local search heuristics). It was found that local optima of evaluations closer to that of the globally optimal evaluation tended to be closer together than local optima of worse evaluations. This lead to the Big Valley Hypothesis which states the search space of the TSP consists of a a single central valley of solutions, at the bottom of which are the global optimum (or optima). In this coarse view, the big-valley structure is conceptualized as a landscape where many local optima may exist, but are easy to escape, and the gradient leads to the global optimum.

This would imply that if an approximation algorithm with a powerful local search heuristic and a method by which to escape local optima, such as Chained-LK, gets close enough to the global optimum, it should almost always find the optimal solution as the gradient leads directly to it. However, we find this not to be the case. The results of the previous chapter indicate that Chained-LK does not find the global optima all the time, in fact, it finds a globally optimal solution in much less than half the trials in most instances tested.

In Table 3.2, Chained-LK was run for 1010 iterations. To further illustrate the point that Chained LK is getting "stuck" we ran the algorithm on the att532 instance for 1,000 trials. In each trial, Chained-LK was allowed to run until the best found tour was not improved for at least 10,000 iterations. As a point of reference, when the global optimum is found, it is often found in as few as 1,000 iterations of Chained-LK. Out of these 1,000 trials, only 4 unique evaluations were found. Of these, one was the global optima which was found 55.9% of the time and the other three tours had evaluations within 0.06% of the optimal evaluation. By the Big Valley hypothesis, the three non-optimal solutions should be very close to the global optimum as they are within only 0.06% of the optimal evaluation. If this were the case, Chained-LK should have found the global optimal solution in all 1,000 trials.

We will first conduct experiments to explore why Chained-LK is getting stuck in a handful of tours that are very close in evaluation to the global optimum, a result counter-intuitive to the Big Valley hypothesis. We will then perform a similar analysis on GPX to determine the cause of its inability to consistently find the global optima. The results of this analysis provide exciting

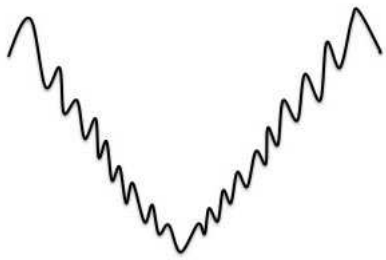


Figure 4.1: A pictorial representation of the big-valley structure.

directions for future research.

4.1 The Search Space of the TSP

In this chapter we discuss structures in the search space of the TSP. It is important to realize that the search space depends on several factors and changing any of these factors can result in an entirely different space. To formalize the concept of search space, we will use the definition of a combinatorial landscape of [RS02], which consists of three ingredients. Specifically for the TSP we have:

1. A set X of all possible solutions. For the symmetric TSP, $|X| = \frac{(n-1)!}{2}$.
2. A notion of distance on X.
3. A fitness function $f : X \mapsto \mathbb{R}$. This is the typical cost function: a summation of the cost associated to each edge in a solution.

For the distance metric, in keeping with the earlier work on the big valley in the TSP, we will use the bond distance of [BKM94a]: Given two tours t_1 and t_2, the bond distance $b(t_1, t_2)$ is equal to n minus the number of edges present in both t_1 and t_2, where n is the number of cities in a particular instance.

In [BKM94a] it was shown that the bond distance $b(t_1, t_2)/2 \leq d(t_1, t_2) \leq b(t_1, t_2)$ where $d(t_1, t_2)$ is the minimum number of 2-opts needed to transform t_1 into t_2. Because LK-search builds a sequence of 2-opt moves that are bounded by the search depth parameter, the bond distance is still meaningful in context of LK-search and is straightforward to compute. Indeed, no polynomial time algorithm is known to compute the minimum number of 2-opt moves required to move from t_1 to t_2 [BKM94a].

4.2 Funnels in the TSP

We conjecture that the reason Chained LK becomes "stuck" at non-optimal solutions is that the big-valley structure breaks down as search approaches the global optimum, making the optimum harder to discover. The clustering of low cost solutions still takes place as stated in the Big Valley

hypothesis, but instead of a single cluster there are a number of clusters of low cost solutions separated by a non-trivial distance.

We term these clusters of solutions *funnels* as they are analogous to the funnels found in energy landscapes [WMW98] associated with Lennard-Jones (LJ) cluster problems. The equation

$$E = 4\epsilon \sum_{i<j} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

is the Lennard-Jones potential and describes the interaction of atoms in an LJ cluster. A solution to an LJ cluster problem is a three dimensional configuration of n atoms, the number of atoms in a cluster is denoted by LJ_n. The optimal solution is a configuration of atoms which minimize the total energy of the system. The evaluation of the Lennard-Jones potential over all possible configurations is known as the potential energy surface, or energy landscape.

In [DMW99], it was observed that some clusters, such as LJ_{55}, have a potential energy surface which contains a single deep funnel which leads to the global minimum while others have more than one funnel. LJ_{38} has two. In the case of LJ_{38}, the bottom of one funnel corresponds to the global minimum, and the bottom of the other funnel corresponds to the configuration with the second lowest energy.

It was found that global optimization is more difficult on multiple funnel landscapes such as LJ_{38} than on larger problems with a single funnel, such as LJ_{55}. This relationship between difficulty in global optimization and the presence of funnels was first shown in [DMW99]. Although the LJ_{38} cluster has been successfully optimized globally, the techniques used [DW98, PP95] involve transformations of the energy landscape which induce a single funnel.

The difficulty that arises in global optimization on energy landscapes with multiple funnels is due to the large barriers between the funnels. These barriers are large enough that once a configuration at the bottom of a funnel is reached, the search is unable to escape the funnel [DMW99]. Thus, once the search enters a funnel, it can be driven to the bottom of the funnel, but if that funnel does not contain the globally optimal configuration, the search will be unable to reach the globally optimal energy configuration.

In the TSP we find that, similar to the behavior of continuous search heuristics on multiple funnel potential energy surfaces, once Chained-LK enters a funnel and reaches a solution corresponding to the bottom of a non-globally optimal funnel, it is highly unlikely that the search will find the global optimum.

In order to further explore funnels in the TSP, we took two approaches to assessing the structure of the search space near the global optimum. First, we examined the number of unique tours found after considerable search effort. We ran Chained-LK until at least 10,000 iterations are performed without finding an improving tour. Chained-LK was run in this manner for 1,000 runs, recording the best tour found from each run. Of the 1,000 best tours produced from each run, we only keep unique tours, i.e., a tour must have at least one different edge from the other tours that are kept. We collected data on 6 instances, rand500, att532, u574, nrw1379, rand1500 and u1817. These instances represent one small and one large instance from each of the three categories of TSP instances. Table 4.1 shows relatively few unique evaluations across the 1,000 runs.

Second, we explored the regions around the unique evaluations. For this analysis, we focus on att532 because it has just four unique evaluations which can be easily displayed graphically.

Instance	Number of unique tours	Number of unique evaluations
Random Euclidean Instances		
rand500	3	2
rand1500	194	151
Clustered or Psuedo-Random Instances		
att532	20	4
nrw1379	902	71
Highly Structured or Grid Instances		
u574	13	4
u1817	967	264

Table 4.1: Number of unique tours and evaluations found after running Chained-LK for 1,000 runs on 10 instances.

Evaluation	27686	27703	27705	27706
27686	-	66	36	86
27703	-	-	76	43
27705	-	-	-	73
27706	-	-	-	-

Table 4.2: Pairwise distance of each funnel bottom used as an initial tour in Figure 4.2. This shows what appears to be an overlap in funnels in Figure 4.2 is not necessarily the case.

For att532, each set of unique tours with the same evaluation formed a connected component in the search space under 2-opt; so the tours are considered to be on plateaus. One tour from each of the four plateaus was selected randomly to represent the funnel bottom (the largest plateau was of size six). 500 local optima were found with the random walk procedure starting from each of the four funnel bottoms. Figure 4.2 plots the distance of the tours to the global optimum[1] versus the tour evaluations; the majority of local optima are further away from the global than the tours associated with the bottoms of each funnel.

These plots concentrate attention on part of the search space very close to the global. The local optima found in each funnel are correlated in evaluation and distance to their respective funnel bottoms and other tours in that funnel. However, a correlation does *not* exist between evaluation and distance for tours from different funnels.

Each funnel itself is a non-trivial bond distance from the other funnels though this is not evident in Figure 4.2. For example, the funnels with bottoms corresponding to plateaus with evaluations of 27705 and 27703 seem to be close to one another in Figure 4.2. However, this is only because we are measuring the distance relative to the global optima, the bond distance between the two tours is actually 76. The pairwise distances between the funnel bottoms are in Table 4.2.

[1] In fact, we found two distinct globally optimal solutions. Because they differ by only two edges, we selected one for distance computation.

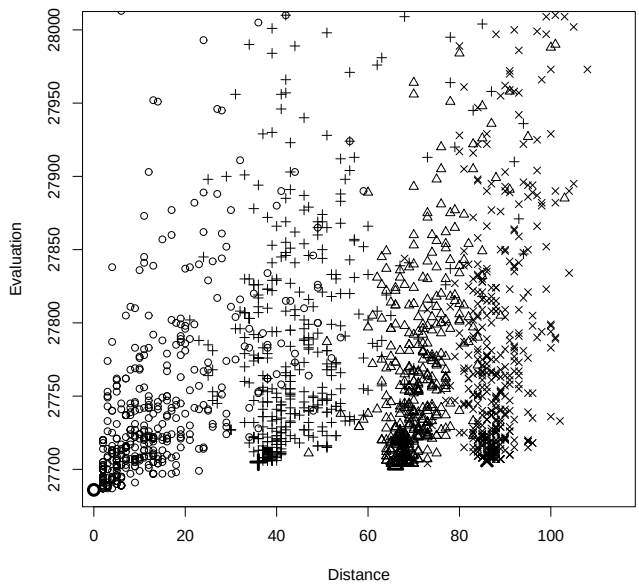


Figure 4.2: 500 local optima found by LK search when starting at each of the four funnel bottoms in att532. Each local optima is shown with a symbol matching the initial tour of the walk. The initial tour, i.e. the funnel bottoms, are shown with a slightly larger and bolder symbol. The initial tours have evaluations from left to right of 27686 (optimal), 27705, 27703 and 27706.

Instance	rand500	att532	u574	nrw1379	rand1500	u1817
Minimum distance	28	26	32	46	54	75

Table 4.3: Minimum bond distance between any two funnels found in each of the instances. The distance between any two funnels must be at least equal to or larger than the numbers reported in this table.

Table 4.2 shows that the funnels corresponding to tours of unique evaluations were always separated by a non-trivial distance for instance att532. We found this to be the case for all instances tested.

Table 4.1 shows that the number of funnels tends to grow with the instance size for all problems. Although impractical to list the pairwise tables of funnel distances for the large instances due to the large number of funnels in these instances, we list the minimum distance between any two funnels in Table 4.3. We can see a general trend for the minimum distance to increase with the size of the instance.

Rather than a single big-valley, there are multiple funnels associated with local optima close in evaluation to the global optimum. Under Chained-LK, the big-valley structure holds within funnels but not across funnels below a certain evaluation.

Instance	Mean bond distance	Mean length of walk
rand500	27.49 ± 1.05	1 ± 0
att532	22.5 ± 1.44	1.01 ± 0.01
u574	24.68 ± 1.57	1.03 ± 0.02
nrw1379	13.74 ± 1.16	1.07 ± 0.03
rand1500	20.04 ± 1.29	1.2 ± 0.04
u1817	17.3 ± 1.14	1.26 ± 0.05

Table 4.4: Mean bond distance from the initial local optima v_0 to the local optima v^* found by LK-search along random double walks from the initial optima and the mean length of the walk.

4.2.1 Implications for Chained-LK Performance

We now return to the conundrum of why local search is not more effective at finding the global optimum in TSP. Chained-LK can move through the search space in two ways: LK-search and double bridge moves. The lowest bond distance between the global optimum and a non-globally optimal solution is 36 for att532 (this distance only grows with instances of larger size).

To determine the mean bond distance a tour is moved by LK-search and the double bridge move, we use a random walk method similar to the one used in Chapter 2 to explore the space of local optima near the global.

For a given instance of the TSP, an initial tour v_0 was found by applying 200 iterations of Chained-LK to a randomly generated tour. We start the walk at tour v_0. At step i of the random walk, a new tour v_i is produced by applying a random double move to v_{i-1}, by breaking four random edges in v_{i-1}, $e(a, b)$, $e(c, d)$, $e(w, x)$ and (y, z), and replacing them with four new edges, $e(a, w)$, $e(b, x)$, $e(c, y)$ and $e(d, z)$, to form the tour v_i. After each step, LK-search is applied to v_i until a locally optimal tour v^* is found. If v^* differs from v_0, v_i has escaped the basin of attraction surrounding v_0; i is said to be length of the walk and the random walk is halted.

When LK-search finds a new local optima v^*, we measure the bond distance between v^* and v^0 and the length of the walk i. The mean distances of 100 such walks are reported in Table 4.4.

The only difference between this random walk method and method used in Chapter 2 is the use of double bridge moves instead of 2-opt moves. The random double bridge move is roughly equivalent to two random 2-opt moves (four edges go out of the tour and four new edges come into the tour in both a double bridge move and two 2-opt moves). The double bridge move does, however, sample a different space than two random 2-opt moves would. If either of the two pairs of edges that are replaced by the double bridge move were broken and replaced individually, a cycle would be formed, making the resulting solution invalid. So, the random double bridge move consists of two *invalid* random 2-opt moves, moves that would not be performed by our random 2-opt walk.

The second column of Table 4.4 tells us that most of the time a single double bridge move was sufficient to find a local optima with LK-search. When comparing the average distance between the new local optima found, in all cases it is smaller than the minimum distance between funnels reported in Table 4.3. Chained-LK applies only a single double bridge move between calls to LK-search and moves to a new local optima only if the evaluation is better than the current best tour. If the search is in a funnel bottom and can only move, on average, a distance smaller than the minimum distance between funnels as indicated by Table 4.4, it is highly unlikely the search

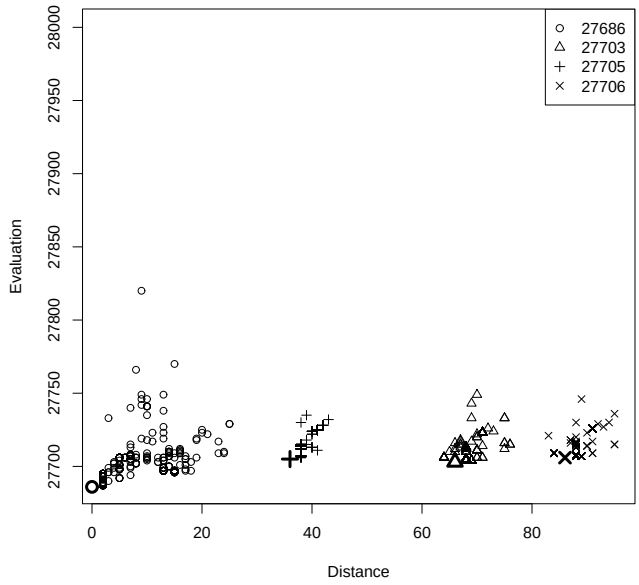


Figure 4.3: 500 local optima found by Lin-Kernighan search after perturbing the four funnel bottoms from att532 by double bridge moves.

will escape the funnel.

To illustrate this idea, we plotted 500 tours in Figure 4.3 as we did in Figure 4.2 using the random double bridge walk instead of the random 2-opt walk. Of the 500 tours found from each initial tour, only a handful of unique local optima were found and they clustered even closer to the initial tours than in Figure 4.2.

The distance between funnels only grows with instance size. Table 4.4 indicates the distance of new local optima found by double bridge moves seems to become less with instance size. As instance size grows, it will become more and more unlikely that LK-search can hop from a non-globally optimal funnel bottom to the globally optimal solution (or a solution in the funnel containing the global optimum) by perturbing a low cost tour using random double bridge moves.

4.3 Escaping Funnels with Generalized Partition Crossover

We now attempt to determine if part of the improved performance of GPX-GA over that of Chained-LK is because the GPX operator escapes funnels by randomly choosing two distinct funnel bottoms from each instance. Only funnels that were not associated with the global optimum were chosen. To generate parent tours associated with a funnel, the random walk method starting from the funnel bottoms was used to find 50 local optima within each funnel. When Chained-LK is applied to these local optima, the search always ends back in the funnel bottom from which the random walk was initiated.

To test the effectiveness of the GPX operator in escaping funnels, we apply the operator to

Instance	rand500	att532	u574
Escapes	1288 (51%)	1129 (45%)	940 (38%)
Instance	nrw1379	rand1500	u1817
Escapes	2254 (91%)	1655 (66%)	1502 (60%)

Table 4.5: Number of funnel escapes out of 2500 crossovers when applying GPX to two tours in different funnels.

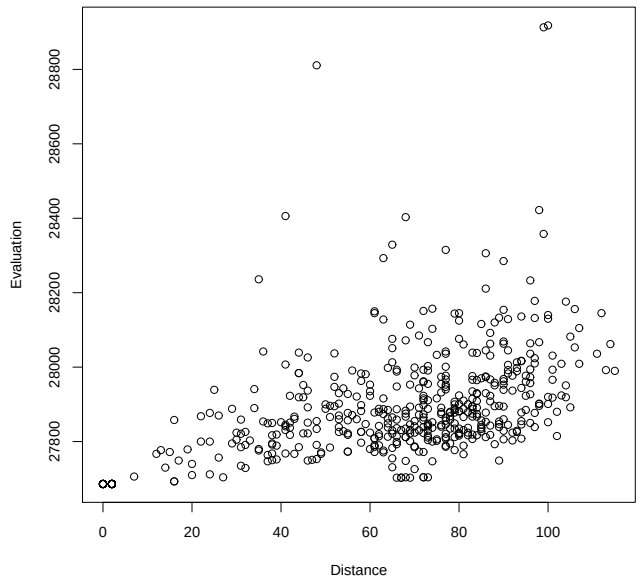


Figure 4.4: Distance to optimal (x-axis) vs. cost function evaluation (y-axis) of the 500 tours produced by the GA using GPX on ATT532 after 10,010 LK search call.

all pairs of tours such that each pair consists of one tour from each of the two funnels. This results in 2,500 applications of the GPX operator per instance. Two offspring are generated, one of which is the result of greedy recombination. Chained-LK is then applied to the two tours which are produced by GPX until it finds no improving tour for at least 10,000 iterations. If one of the tours found by Chained-LK is in a different funnel from the initial funnels, we conclude that a single application of the GPX operator was successfully able to escape the initial funnels.

The results are presented in Table 4.5. In all instances the GPX operator displayed the ability to produce a tour that led to a different funnel than the funnel in which the parents were found. GPX is able to tunnel between funnels, something that LK-search and double bridge moves alone cannot.

We also looked at the results of running GPX-GA with a population of 10 tours until 10,000 LK-seach calls were made. The results of doing this for 100 trials are depicted in Figure 4.4 using the same cost versus distance plot as in Figure 4.3 except that these are the tours found by GPX-GA at the end of each run.

The clustering found in Figure 4.3 is not evident in Figure 4.4 and Figure 4.4 displays a more even distribution of tours over the lower part of the search space. This seems to indicate that GPX-GA is not susceptible to the funnel structures found when using Chained-LK. The search space configuration is altered by the use of GPX-GA, similar to the transformation methods used in continuous optimization to avoid funnels. This then leaves the question, why does GPX not find the global optimum more frequently?

4.4 What Makes Generalized Partition Crossover Fail

We analyzed the populations in the final generations of the experiments in Chapter 3 in which the global optimum *was not* found after GPX-GA had been run for 100 generations with a population of size 10. We found that *all* the edges found in the globally optimal tour were found in the edges of the entire population, albeit not in the best found tour, in the majority of trials for all instances tested. Why then is GPX unable to recombine tours to find the global optimum if the edges are present?

Recall that in GPX-GA we recombine the best tour with the remaining tours in the population. Those globally optimal edges that are not found in the best tour are found somewhere in at least one other tour in the population, therefore they will be classified as uncommon edges in at least one recombination.

To ensure that GPX always produces a valid Hamiltonian circuit when constructing offspring, edges cannot be selected on an individual basis within partition components. That is, the edges used to construct a child tour from within a particular partition component must all come from one parent or the other. During recombination, if some globally optimal edges which are found in one parent appear in the same partition component as *different* globally optimal edges from the other parent, it will be impossible for GPX to recombine the two tours and find the globally optimal edges as it cannot construct a child tour using the edges from both parents within a single partition component.

Because of this we wish to determine whether the globally optimal edges that aren't found in the best tour appear in a single component during recombination or if they are spread out among different components. We checked this by running GPX-GA for 100 generations and analyzing each component of every recombination that produced the final generation.

We found that in all the recombinations prior to the final 100th generation, the majority of globally optimal edges fell into a single partition component that was larger than the rest. In table 4.6 we report the number of uncommon globally optimal edges that fell into this large partition component, the size of this component, and the number of uncommon global edges that are found in other components; we also report the number of shared common global edges observed during recombination.

As can be seen from Table 4.6, the majority of edges that are not found in the best solution but that are found in the global optimal solution appear as uncommon edges that are largely contained in the largest component during the recombination process. For this reason GPX is unable to recombine tours to find the global optimum.

As a final experiment, we examine how quickly the global edges are found in the population. The results of this experiment lead to speculation on a different strategy for using GPX-GA to avoid the problem of recombining to find the global optimum. We turned off the double bridge

	rand500	att532	u574
Common global edges	392.39	406.75	440.86
Uncommon global edges in largest component	50.5	87.39	56.15
Uncommon global edges not in largest component	0.04	0.04	0.81
Total edges in the largest component	67.17	102.11	80.15
	nrw1379	rand1500	u1817
Common global edges	995.52	1120.50	1407.90
Uncommon global edges in largest component	160.78	171.47	268.62
Uncommon global edges not in largest component	1.80	1.05	0.96
Total edges in the largest component	222.83	223.86	289.12

Table 4.6: Percentages were averaged over 50 trials. These results were captured during recombination during the last generation. Common edges can appear inside of components, or between components of Graph G_u. Uncommon edges appear only inside of components of G_u.

mutation in step 2 of the GPX-GA as described in Figure 3.3 in Chapter 3. Without double bridging, the population converges quickly, and we only run the GA to 5 generations.

We ran 50 trials of the modified GPX-GA. At generation 5 we record the minimum tour found, the number of unique edges in the population and the number of edges in the population that also appear in the global optimum. When there is no mutation (i.e., double bridge moves) not only does the GA converge very fast, but it also loses diversity and gets "stuck" after about 5 generations.

Nevertheless, GPX-GA is already finding very good solutions after only 5 generations: for rand500 and att532, it found the global optimum in 2 out of 50 trials, using only 50 recombinations and only 50 calls to LK-search. The convergence to the global optimum is extremely fast in these exceptional cases.

As the data show in table 4.7, the edges found in the global optimum are all present in the population in the majority of the runs. On instances rand500, att532 and rand1500 all of the globally optimal edges were present in the population on every single run. The population therefore contains all of the edges needed to construct the globally optimal solution after only 5 generations. Furthermore, the results for all of the TSP instances show that the total number of unique edges in the population was always less than $2n$ after 5 generations.

These results suggest a different use of GPX-GA: using GPX-GA to quickly create a population of tours to be used in a strategy known as *tour merging*. Tour merging [App06, CS03] is a strategy which involves merging together a number of tours to create a graph which is then passed to an exact solver for the TSP. This merging is similar to what happens when taking the union of two tours in partition crossover, but typically involves more than two tours. The end result is a graph with a greatly reduced search space than the complete graph of a TSP instance and can , theoretically, be searched in a much shorter time. We will discuss this and other future work in the next chapter.

Instance	Global Edges in Population	Global Edges in Minimum Tour	Unique Edges in Population
rand500	500 ± 0	449.68 ± 1.98	941.56 ± 1.56
att532	532 ± 0	464.1 ± 2.11	979.54 ± 1.47
u574	574 ± 0	502.63 ± 2.58	1017.12 ± 1.87
nrw1379	1378.9 ± 0.04	1162.3 ± 3.44	2709.34 ± 2.25
rand1500	1500 ± 0	1301.02 ± 4.15	2871.9 ± 3.14
u1817	1815.12 ± 0.18	1562.44 ± 3.22	3616.92 ± 4.71

Table 4.7: Results obtained by running GPX-GA for only 5 generations and *without* mutation (double-bridge moves). "Global Edges in Population" is the number of edges found in the global optimum that are also present in the population. "Global Edges in Minimal Tour" is the number of edges shared in common between the best solution found and the global optimum. "Unique Edges in Population" is the total number of distinct edges found in the population at the end of generation 5. The number of edges in each instance is indicated by the numerical suffix of the instance name.

Chapter 5

Conclusions

In this work we have presented a new recombination operator called Generalized Partition Crossover /citewhitley-hybrid for the Traveling Salesman Problem, designed a hybrid genetic algorithm, which outperforms Chained-LK, and conducted analyses revealing search space structures that help explain the difference in performance.

GPX makes use of multiple partitions of cost two on the union graph of two initial tours. The number of partitions of cost two are found to affect the performance of the operator, namely there is correlation between the number of partitions found by the operator and the number of offspring that are improved over the parents. Although the operator will be unsuccessful if no partitions of cost two are present, we have shown that the operator achieves a high rate of successful recombinations.

We have found that GPX possesses several interesting characteristics, namely: it transmits alleles, it is respectful and it exhibits a property we refer to as tunneling. These characteristics play an important role in designing GPX-GA. Because the operator transmits alleles and is respectful, it cannot introduce new edges into a population of solutions. We therefore utilize a diverse selection method and the double bridge move as a mutation operator to increase diversity in the population. Tunneling occurs when the parents are both local optima, and the quality of the offspring of GPX are shown to be correlated to the quality of the parents. We therefore incorporated the use of LK-search, the most powerful local search heuristic known for the TSP, into GPX-GA.

The end result is an algorithm that is capable of outperforming Chained LK, one of the best performing approximation algorithms for the TSP as of this writing. Analyses of why Chained-LK becomes stuck reveal search space structures that we call funnels after a similar phenomenon found in energy landscapes. Funnels are tours in the search space that are correlated in distance and evaluation, but are themselves separated from one another by a non-trivial distance.

A gradient is formed within funnels that draws Chained-LK to a funnel bottom and, once there, it is highly unlikely to escape to another funnel. Because GPX is not limited to changing a small number of edges as is the double bridge move, the search space at low evaluations is smoother out and the funnel structures are not as evident. We conjecture that this is in part a reason why GPX is capable of outperforming Chained LK.

Having conducted an analysis on why Chained-LK fails, we perform a similar analysis on GPX-GA. We find that when GPX-GA fails to find the global optimum, it is because the globally optimal edges are spread out among different tours but fall into the same partition component

during recombination. This makes it impossible for GPX to recombine two tours to find the global optima.

We also find that the globally optimal edges are entirely present in the population after only five generations in the majority of trials. For some instances, they were present in every trial we ran. These final experiments give rise to several ideas for future research directions involving GPX.

5.1 Future Directions for GPX

One direction for future research is investigating ways in which GPX can select edges from both parents within a single partition component. We have shown that the globally optimal edges are contained within the largest partition component in many cases. One possibility is allowing the algorithm to run until GPX is unable to recombine to form better tours. At this point, GPX can be used to find the partition components but instead of constructing offspring at this point, a search algorithm may be applied to each component, treating each partition component as a sub-problem.

As each partition component has a partition cost of 2 (after the recursive cut point reduction step) there are two easily identified cut points at which any Hamiltonian circuit on the entire problem must enter and exit the partition component. This means the component can be optimized independently of the rest of the problem and all the necessary information to do so is provided from GPX with no extra work.

It may also be possible to perform a different type of recombination in which the edges are selected one by one in such a way as to not form a cycle. We obviously know that there is at least one other way to form a path through this component, the globally optimal path. However, this problem does seem to be difficult to solve without increasing the running time of the operator.

Other influences may be applied during the search, such as the backbone search of [ZL05], before GPX-GA reaches a point where it is incapable of recombining to find the global. From our analysis, we know GPX is sometimes able to recombine two tours to find the global optimum and GPX-GA does so in many cases. A study of the search trajectories taken from the first to last generation may yield some insight into what types of influences may be applied to guide the search to favorable tours. This may not be a straight forward undertaking as the population of GPX-GA makes such a study more complicated than an approximation algorithm which maintains only a single tour.

Perhaps the most exciting possibility is running GPX-GA for only a few generations and merging the tours in the resulting population. Our results show that in as few as five generations the globally optimal edges were represented across the population in the majority of instances. GPX-GA could be run with mutation (double bridge moves) turned off until convergence. We can then merge all 10 members of the population into a single graph. This reduced graph can then be searched for a minimal Hamiltonian circuit using an exact algorithm which can take advantage of the reduced space, such as those described in [App06] and [CS03].

The search space is dramatically smaller than that of the original TSP instances. This means that the optimization problem has been reduced to finding the minimal Hamiltonian Circuit of length n in a graph with at most kn edges where k is the size of the population. Such a strategy utilizes the ability of GPX-GA to quickly produce a population of diverse, high quality tours. These tours may then be used to construct an instance which can be turned over to an exact

solver. If the optimal edges are present, the solver will have a greatly reduced space to explore and will still be able to find the optimal solution.

REFERENCES

[ABCC98] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. On the solution of traveling salesman problems. *Documenta Mathematica*, 3:645–656, 1998.

[ABCC03] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. *Mathematical Programming*, 97(1):91–153, 2003.

[ACR03] D. Applegate, W. Cook, and A. Rohe. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003.

[App06] D.L. Applegate. *The traveling salesman problem: a computational study*. Princeton Univ Pr, 2006.

[BKM94a] K.D. Boese, A.B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16(2):101–114, 1994.

[BKM94b] Kenneth D. Boese, Andrew B. Kahng, and Sudhakar Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16:101–113, 1994.

[Bra91] H. Braun. On solving travelling salesman problems by genetic algorithms. *Parallel Problem Solving from Nature*, pages 129–133, 1991.

[BS02] H.G. Beyer and H.P. Schwefel. Evolution strategies–A comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

[CG76] N. Christofides and CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEARCH GROUP. *Worst-case analysis of a new heuristic for the travelling salesman problem*. Management Science [s] Research Group, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[Cro58] GA Croes. A method for solving traveling-salesman problems. *Operations Research*, pages 791–812, 1958.

[CS03] W. Cook and P. Seymour. Tour merging via branch-decomposition. *INFORMS Journal on Computing*, 15(3):233–248, 2003.

[DMW99] J.P.K. Doye, M.A. Miller, and D.J. Wales. The double-funnel energy landscape of the 38-atom Lennard-Jones cluster. *The Journal of Chemical Physics*, 110:6896, 1999.

[DW98] J.P.K. Doye and D.J. Wales. Thermodynamics of global optimization. *Physical Review Letters*, 80(7):1357–1360, 1998.

[FM96] B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. *Parallel Problem Solving from Nature.PPSN IV*, pages 890–899, 1996.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences*. WH Freeman and Company, San Francisco, Calif, 1979.

[GS02] M. Gorges-Schleuter. Asparagos96 and the traveling salesman problem. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 171–174. IEEE, 2002.

[Hel00] K. Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[Hol92] J.H. Holland. *Adaptation in natural and artificial systems*. MIT press Cambridge, MA, 1992.

[JM97] D. S. Johnson and L. A. McGeoch. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons Ltd, 1997.

[KT85] S. Kirkpatrick and G. Toulouse. Configuration space analysis of travelling salesman problems. *Journal de Physique*, 46(8):1277–1292, 1985.

[LK73a] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[LK73b] S. Lin and BW Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, pages 498–516, 1973.

[LLKS85] E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, and D.B. Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley New York, 1985.

[MF97] P. Merz and B. Freisleben. Genetic local search for the TSP: New results. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 159–164. Citeseer, 1997.

[MFMS99] A. Möbius, B. Freisleben, P. Merz, and M. Schreiber. Combinatorial optimization by iterative partial transcription. *Physical Review E*, 59(4):4667–4674, 1999.

[MGSK88] H. Muhlenbein, M. Gorges-Schleuter, and O. Kramer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7(1):65–85, 1988.

[Nag97] Y. Nagata. Edge Assembly CrossoveriA High-power Genetic Algorithm for the Traveling Salesman Problem. In *Proceedings of the Seventh International Conference on Genetic Algorithms, Michigan State University, East Lansing, MI, July 19-23, 1997*, page 450. Morgan Kaufmann Publishers, 1997.

[Nag06] Y. Nagata. New EAX crossover for large TSP instances. *Parallel Problem Solving from Nature-PPSN IX*, pages 372–381, 2006.

[NK97] Yuichi Nagata and Shigenobu Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In T. Bäck, editor, *Proc. of the 7th Int'l. Conf. on GAs*, pages 450–457. Morgan Kaufmann, 1997.

[PP95] J. Pillardy and L. Piela. Molecular dynamics on deformed potential energy hypersurfaces. *The Journal of Physical Chemistry*, 99(31):11805–11812, 1995.

[RS95] N.J. Radcliffe and P.D. Surry. Fitness variance of formae and performance predictions. In D. Whitley and M. Vose, editors, *FOGA - 3*, pages 51–72. Morgan Kaufmann, 1995.

[RS02] C.M. Reidys and P.F. Stadler. Combinatorial landscapes. *SIAM review*, pages 3–54, 2002.

[TM98] G. Tao and Z. Michalewicz. Inver-over operator for the TSP. In *Parallel Problem Solving from NaturePPSN V*, page 803. Springer, 1998.

[Vos99] M.D. Vose. *The simple genetic algorithm: foundations and theory*. The MIT Press, 1999.

[WHH] D. Whitley, D. Hains, and A. Howe. A Hybrid Genetic Algorithm for the Traveling Salesman Problem Using Generalized Partition Crossover. *Parallel Problem Solving from Nature–PPSN XI*, pages 566–575.

[WHH09] D. Whitley, D. Hains, and A. Howe. Tunneling between optima: partition crossover for the traveling salesman problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 915–922. ACM, 2009.

[Whi94] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[WMW98] D.J. Wales, M.A. Miller, and T.R. Walsh. Archetypal energy landscapes. *Nature*, 394(6695):758–760, 1998.

[WRE+98] J. Watson, C. Ross, V. Eisele, J. Denton, J. Bins, C. Guerra, D. Whitley, and A. Howe. The traveling salesrep problem, edge assembly crossover, and 2-opt. In *Parallel Problem Solving from Nature.PPSN V*, page 823. Springer, 1998.

[WSF89] Darrell Whitley, Timothy Starkweather, and D'ann Fuquay. Scheduling problems and traveling salesmen. In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[ZL05] W. Zhang and M. Looks. A novel local search algorithm for the traveling salesman problem that exploits backbones. In *International Joint Conference on Artificial Intelligence*, volume 19, page 343. Citeseer, 2005.