Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Optimization 3 (2006) 78–85

DISCRETE OPTIMIZATION

www.elsevier.com/locate/disopt

A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem

Özlem Ergun[a,*], James B. Orlin[b]

[a] Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30339, United States
[b] Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, United States

Received 13 December 2004; received in revised form 21 August 2005; accepted 17 October 2005
Available online 19 January 2006

Abstract

We consider the standard dynamic program to solve the TSP. We then obtain exponentially large neighborhoods by selecting a polynomially bounded number of states, and restricting the dynamic program to those states only. We show how the Balas and Simonetti neighborhood and the insertion dynasearch neighborhood can be viewed in this manner. We also show that one of the dynasearch neighborhoods can be derived directly from the 2-exchange neighborhood using this approach.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Traveling salesman problem; Very large-scale neighborhood search; Dynamic programming

1. Introduction

Neighborhood search is a practical method for efficiently finding "good" solutions to hard combinatorial optimization problems. Let $P = \min\{cx : x \in D\}$ be an instance of an optimization problem with cost vector c and feasible set D. Given a feasible solution x in D, a neighborhood search algorithm has an associated neighborhood function N that identifies a subset, $N(x)$, of D as the "neighbors" of x under N.

A local search algorithm has the following basic scheme. Start with a current feasible solution, say x, and iteratively replace the current solution with a neighbor y of the current solution with lower objective value. Continue until there is no neighbor with improved objective value, at which point the current solution is called *locally optimal*. There is a large literature on local search as well as extensions of local search including simulated annealing and tabu search. For an excellent reference on local search in combinatorial optimization, see [1].

In very large-scale neighborhood (*VLSN*) search, the number of solutions in a neighborhood is very large (often exponential) with respect to the size of the input. As a rule of thumb, one expects to find better locally optimal solutions assuming that one can search a larger neighborhood efficiently. Unfortunately, for many very large neighborhoods, the search time may be much larger. There are a variety of techniques for efficiently searching neighborhoods in *VLSN* search. One general approach that has been successful in searching exponentially large neighborhoods in polynomial

* Corresponding author.
E-mail addresses: oergun@isye.gatech.edu (Ö. Ergun), jorlin@mit.edu (J.B. Orlin).

1572-5286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2005.10.002

time has been dynamic programming. See [3,12,9] for surveys on these techniques, including a number of papers on *VLSN* search that employ dynamic programming. See also [6] for representing neighborhoods using context free grammars so that the dynamic program is derived automatically from the grammar.

In this note, we present a simple approach for creating *VLSN*s that are searchable in polynomial time. Our approach starts with the standard dynamic program to solve a combinatorial optimization problem and restricts attention to a polynomially large subset V of states of the dynamic programming state space. When $|V|$is polynomially bounded in the size of the input, the time to solve the dynamic program is guaranteed to be polynomial. We give examples in which $|V|$ is polynomially bounded, and solving the dynamic programming recursion over V is equivalent, in a technical sense that we will make clear, to searching an exponentially large neighborhood. We illustrate this approach on dynasearch neighborhoods and extensions [14,7] as well as on the Balas–Simonetti neighborhood [5]. Furthermore, we provide a method of using a dynamic programming recursion to transform a neighborhood N into a possibly larger neighborhood N' that is called the "dynamic programming expansion" of N. In the case of the 2-exchange neighborhood, the dynamic programming expansion is exponentially large, but can be searched in polynomial time.

2. Definitions and background for the TSP

In this section we offer definitions and background for the symmetric (equivalently undirected) Traveling Salesman Problem (TSP).

2.1. The traveling salesman problem

The traveling salesman problem tries to find the minimum distance tour on *n cities* that are labeled $1, 2, \ldots, n$. Let the distance from city i to city j be $c(i, j)$. Let S_n denote the set of all permutations of $\{1, 2, \ldots, n\}$. Then given a permutation $T \in S_n$, with $T = T(1), T(2), T(3), \ldots, T(n)$, a unique *tour*, $\tau(T)$, can be associated with permutation T. The tour $\tau(T)$ refers to the tour in which city $T(2)$ is visited between cities $T(1)$ and $T(3)$, and city $T(n)$ is visited between cities $T(n-1)$ and $T(1)$, and so on. Note that there are n distinct permutations associated with the same tour. We refer to a pair of consecutive cities of $\tau(T)$ (including $T(n), T(1)$) as an *edge* of the tour $\tau(T)$. We denote a sequence A of k cities as $A = \langle i_1, i_2, \ldots, i_k\rangle$, and we let $Rev(A) = \langle i_k, i_{k-1}, \ldots, i_1\rangle$. If $i \leq j$, we let $[i, j]$ be shorthand for the sequence $\langle i, i+1, \ldots, j\rangle$. If $i > j$, then $[i, j] = \phi$. The subset obtained from subset S of cities by deleting all cities in S' will be denoted as $S \setminus S'$. We abbreviate $S \setminus \{i\}$ as $S \setminus i$. If S is a subset or sequence of cities, then $\max(S)$ denotes the maximum index of a city of S, and $\min(S)$ denotes the minimum index of a city of S.

The cost of a sequence $A = \langle i_1, i_2, \ldots, i_k\rangle$ of cities is $c(A) = \sum_{j=1}^{k-1} c(i_j, i_{j+1})$. If $k = n$, then the cost of the associated tour $\tau(A)$ is $c(\tau(A)) = c(A) + c(i_n, i_1)$.

As per Deĭneko and Woeginger [9], if A is a sequence of cities, and B is a different sequence of cities with no city in common with A, then $A \star B$ is the sequence obtained by concatenating A with B. For example, $\langle 3, 1, 7\rangle \star \langle 2, 6, 4\rangle = \langle 3, 1, 7, 2, 6, 4\rangle$.

2.2. Notation for neighborhoods for the traveling salesman problem

In subsequent sections, we assume that we are starting neighborhood search starting with the tour $T^I = \langle 1, \ldots, n\rangle$, and we provide dynamic programs for determining the minimum distance neighbor of $\tau(T^I)$. If one wants to search $N(\tau(T))$ instead, then it suffices to replace c by f_T in the recursions, where f_T is obtained from c by appropriately permuting rows and columns of the distance matrix c.

When the number of cities n is permitted to vary, we let N_n denote the neighborhood set for problems with n cities. In the case that the number of cities is obvious from context, we drop the index, and denote the neighborhood set as N. We assume that the tour associated with the identity permutation $\tau(T^I)$ is in N_n, that is $\tau(\langle 1, 2, \ldots, n\rangle) \in N_n$ for all n.

All of our recursions define sequences of fewer than or equal to n cities. We let N^* refer to all sequences defined by the recursion, and we let $N = \tau(N^* \cap S_n)$ denote the neighborhood. We will refer to the set N^* as a *superneighborhood.*

3. Optimization over DP restrictions

The Held and Karp [13] dynamic program for the TSP can be viewed as being based on the following recursion:

The Complete Neighborhood N_{TSP} for the TSP.
1. $1 \in N^*_{TSP}$.
2. If $A \in N^*_{TSP}$ and $k \notin A$, then $A \star k \in N^*_{TSP}$.
3. $N_{TSP} = \tau(N^*_{TSP} \cap S_n)$.

The state space graph corresponding to the dynamic program for the *TSP* can be obtained as follows. It is a directed graph that includes a node for each state of the dynamic program as well as a special destination node t.

State Space Graph $G_{TSP} = (V_{TSP}, E_{TSP})$.
1. $(\{1\}, 1) \in V_{TSP}$ and $t \in V_{TSP}$.
2. If $(S, j) \in V_{TSP}$, and if $k \notin S$, then $(S \cup \{k\}, k) \in V_{TSP}$, and there is an arc from (S, j) to $(S \cup \{k\}, k)$ in E_{TSP} with cost $c(j, k)$.
3. If $(S, j) \in V_{TSP}$ and if $|S| = n$, then there is an arc from (S, j) to t with cost $c(j, 1)$.

Let $g_{TSP}(S, j)$ denote the optimal value for state $(S, j) \in V_{TSP}$ in the Held and Karp dynamic program for the TSP. Then $g_{TSP}(S, j)$ is the shortest length of a sequence of cities whose initial city is 1, whose terminal city is j, and that includes all of the cities of S.

The following is a well known property of the state space graph and its relation to the Held and Karp dynamic program DP_{TSP}: if we let (S_j, i_j) denote the j-th node of V_{TSP} on a shortest path from $(\{1\}, 1)$ to t, then a shortest length tour is induced by the sequence $\langle i_1, \dots, i_n \rangle$.

In this section and in the next section, we apply dynamic programming restrictions; that is, we take the Held–Karp dynamic program for the traveling salesman problem and solve this dynamic program as restricted to a subset $V \subseteq V_{TSP}$, which is denoted as $G_{TSP}[V]$.

In order to associate neighborhoods with subsets of states, we need some additional notation and definitions. For a given sequence $A = \langle i_1, i_2, i_3, \dots, i_k \rangle$, with $i_1 = 1$, we let $State(A) = (S, i_k)$, where $S = \{i_1, i_2, \dots, i_k\}$. For a sequence $A = \langle i_1, i_2, i_3, \dots, i_k \rangle$, we refer to the subsequences $A_j = \langle i_1, i_2, i_3, \dots, i_j \rangle$ for $j = 1$ to k as the *initial subsequences* of A. The canonical dynamic program creates a tour A by starting with city 1 and concatenating one city at a time. With this in mind, for each tour A we let

$$V_{TSP}[A] = \{t\} \cup \{State(A_j) : A_j = \langle 1, \dots, i_j \rangle \text{ is an initial subsequence of } A \text{ for } j = 1 \text{ to } n\}.$$

For a given collection $V \subseteq V_{TSP}$, let $N_{TSP}[V] = \{\tau(A) : A \in S_n \text{ and } V_{TSP}[A] \subseteq V\}$. In other words, $N_{TSP}[V]$ contains all tours $\tau(A)$ such that the states needed to generate A are all in V. Similarly, we let $N^*_{TSP}[V] = N_{TSP}[V] \cup \{A : V_{TSP}[A] \subseteq V\}$.

The following theorem states that finding the shortest path from node $(\{1\}, 1)$ to all other nodes in $G_{TSP}[V]$ corresponds to finding the best tour in $N_{TSP}[V]$. It is straightforward and stated without proof.

Theorem 1. *Let V be any subset of states of V_{TSP}. Let (S, j) be any state in V, and let $g(S, j)$ be the minimum cost of a path from $(\{1\}, 1)$ to (S, j) in $G_{TSP}[V]$. Then*

$$g(S, j) = \min\{c(A) : State(A) = (S, j) \text{ and } A \in N^*_{TSP}[V]\}, \text{ and}$$
$$g(t) = \min\{c(T) : T \in N_{TSP}[V]\}.$$

3.1. The Balas–Simonetti neighborhood

Balas [4] considered the neighborhood consisting of all sequences A with the following properties: (1) the first city of A is city 1, and (2) there is a parameter K of the neighborhood, such that i follows j in A if $j + K \le i$. We consider the following equivalent formulation: If i precedes j in A, then $i < K + j$. Balas presented a dynamic programming recursion for finding the minimum distance tour in the neighborhood that runs in $O(K^2 2^K n)$ time, which is linear in n for fixed K and polynomial in n for $K = O(\log n)$. The number of tours in the neighborhood is $(\Omega(k-1)^n/e^n)$, which is exponential in n whenever k is a constant larger than e or if k is slowly growing in n.

Balas and Simonetti [5] generalized the neighborhood to require a condition that is equivalent to the following: If i precedes j in A, then $i < K(j) + j$, where $K(j)$ is a positive integer bounded above by K. They also showed how to implement the dynamic program effectively by searching the state space graph, and they carried out extensive computational experiments. Here we show that the Balas–Simonetti neighborhood can be constructed in a very natural way using recursion. Moreover, the recursion immediately suggests an implementation very similar to the one that Balas and Simonetti applied. In the following, if A is a sequence of cities or a set, we let $\bar{A} = \{1, \dots, n\} \setminus A$, that is the cities not in A.

The Balas–Simonetti neighborhood N_{BS} for the TSP.

1. $1 \in N^*_{BS}$.

2. Suppose $A \in N^*_{BS}$ and choose $i' \in \bar{A}$ so that $i' + K(i') = \min\{i + K(i) : i \in \bar{A}\}$. Then for each $j \in \bar{A}$ with $j < i' + K(i')$, $A \star j \in N^*_{BS}$.

3. $N_{BS} = \tau(N^*_{BS} \cap S_n)$.

State Space Graph $G_{BS} = (V_{BS}, E_{BS})$:

1. $(\{1\}, 1) \in V_{BS}$.

2. Suppose $(S, k) \in V_{BS}$, and choose $i' \in \bar{A}$ so that $i' + K(i') = \min\{i + K(i) : i \in \bar{S}\}$. Then for each $j \in \bar{S}$ with $j < i' + K(i')$, $(S \cup \{j\}, j) \in V_{BS}$, and there is an arc from (S, k) to $(S \cup \{j\}, j)$ with cost $c(k, j)$.

3. If $(S, j) \in V_{BS}$ and if $|S| = n$, then there is an arc from (S, j) to t with cost $c(j, 1)$.

We next show that N^*_{BS} is the Balas–Simonetti neighborhood.

Lemma 1. *The tour $T \in N_{BS}$ if and only if whenever i precedes j in T, then $i < K(j) + j$.*

Proof. Suppose $T = \tau(A)$, and $A = \langle 1, i_2, i_3, \dots, i_n \rangle$. Let $A_j = \langle 1, i_2, i_3, \dots, i_j \rangle$. Let us assume first that $T \in N_{BS}$, and thus $A_k \in N^*_{BS}$ for each k. It follows directly from the construction of A_k in Step 2 of the N_{BS} neighborhood, that $i_k < i_l + K(i_l)$ for all $l > k$.

We next assume that whenever i precedes j in A, then $i < K(j) + j$. We will claim that $A_k \in N^*_{BS}$ for all k, and thus $\tau(A) \in N_{BS}$. The claim is clearly true for $k = 1$, and we assume inductively that the claim is true for $k - 1$. By inductive hypothesis $A_{k-1} \in N^*_{BS}$, and by our choice of A, $i_k < i_l + K(i_l)$ for all $l > k$. It follows that $A_k \in N^*_{BS}$, completing the proof. □

The recursion for N_{BS} adds one city at a time to the end of a sequence A such that the newly added city satisfies a condition. Hence the recursion for N_{BS} is a restriction of the Held and Karp recursion for N_{TSP} and $V_{BS} \subseteq V_{TSP}$. Furthermore, by Theorem 1 finding the shortest path from node $(\{1\}, 1)$ to node t in $G_{TSP}[V_{BS}]$ corresponds to finding the best tour in $N_{TSP}[V_{BS}]$ that is equivalent to N_{BS}. Our approach leads to an alternative proof of the following result by Balas and Simonetti [5].

Lemma 2. *Consider the Balas–Simonetti neighborhood and suppose that $K < \log n$. Then $|V_{BS}| = O(nK2^k)$, and $|E_{BS}| = O(nK^2 2^K)$. The time to construct G_{BS} is $O(nK^2 2^K)$, and the time to find a minimum distance neighbor is $O(nK^2 2^K)$.*

3.2. The weak insertion dynasearch neighborhood

A class of exponentially sized neighborhoods was developed in [14,7,8] from the 2-exchange neighborhood by compounding sets of independent 2-exchanges.

We let $h(i, j)$ be the cost associated with sequence $[i, j] = \langle i, i+1, \ldots, j\rangle$. Thus $h(i, j) = \sum_{k=i}^{j-1} c(k, k+1)$. One can first compute $h(1, j)$ and $h(j, n)$ for all j in $O(n)$ steps. Subsequently computing $h(i, j) = h(1, n) - h(j, n) - h(1, i)$ takes $O(1)$ additional steps.

For $i \leq j$, let *Rev*$[i, j]$ denote the sequence $\langle j, j-1, \ldots, i\rangle$. We let *RevMove*$[i, j]$ be the move that reverses the orders of cities in positions i to j. For example, if we apply *RevMove*$[i, j]$ to the identity permutation T^I, we obtain $\langle 1, \ldots, i-1\rangle \star$ *Rev*$[i, j] \star \langle j+1, \ldots, n\rangle$. If we apply *RevMove*$[3, 4]$ to $\langle 1, 5, 2, 3, 4\rangle$, we would obtain $\langle 1, 5, 3, 2, 4\rangle$.

A *2-exchange* of a tour $T = \tau(A)$ is a tour obtained from permutation A by the operation *RevMove*$[i, j]$ for some $i < j$. Equivalently, the 2-exchange of the tour $\tau(\langle 1, \ldots, n\rangle)$ can be viewed as breaking edges $(i-1, i)$ and $(j, j+1)$ and adding edges $(i-1, j)$ and $(i, j+1)$. We say that two 2-exchanges *RevMove*$[i_1, j_1]$ and *RevMove*$[i_2, j_2]$ are *independent* if $j_1 < i_2 - 1$ or $j_2 < i_1 - 1$. If $j_1 < i_2$ or $j_2 < i_1$, we say that the two 2-exchanges are *weakly independent*. Potts and van de Velde [14] and Congram et al. [8] introduced neighborhoods based on compounding (or applying) independent moves that do not involve the first city of the permutation T under the name "dynasearch", a term that we use here as well. We refer to the neighborhood obtained from T^I by applying a compounded set of independent 2-exchanges that do not involve city 1 as the *2-exchange dynasearch neighborhood*. We refer to the neighborhood obtained by applying a compounded set of weakly independent 2-exchanges that do not involve city 1 as the *weak 2-exchange dynasearch neighborhood*.

The size of the compounded independent moves neighborhood is $\Omega(1.7548^n)$, as may be computed from a simple recursion. See [7,10] for derivations. The dynasearch neighborhoods can be searched in $O(n^2)$ by dynamic programs as in [14,7,8], and by network flows techniques as in [2,10,11].

In this section, we show that the weak insertion dynasearch neighborhood N_{WIDS} can be searched efficiently in $O(n^2)$ when described with a recursion that is a restriction of the Held and Karp recursion for N_{TSP}. First we provide a characterization of the weak independent insertion neighborhood, and then give the recursive description.

Lemma 3. *Let $A = \langle 1, i_2, i_3, \ldots, i_n\rangle$, and let $A_j = \langle 1, i_2, i_3, \ldots, i_j\rangle$ for each $j = 2$ to n. Then $\tau(A) \in$ Weak insertion dynasearch neighborhood if and only if for each j, the cities of A_j are $\{1, 2, \ldots, j+1\} \setminus k$ for some $k \in \{2, \ldots, j+1\}$.*

Proof. Suppose first that $\tau(A)$ is in the weak insertion dynasearch neighborhood. If $A = \langle 1, 2, \ldots, n\rangle$, then the cities of A_j are $\{1, \ldots, j\}$, and the lemma is valid. Suppose instead that $A \neq \langle 1, 2, \ldots, n\rangle$, and let *InsertMove*$(r, s)$ be the last InsertMove in the creation of A. We assume inductively that for $j < r$, the cities of A_j are $\{1, 2, \ldots, j+1\} \setminus k$ for some $k \in \{2, \ldots, j+1\}$. Because A is formed by compounding weakly independent moves, the cities of A_{r-1} are $\{1, 2, \ldots, r-1\}$. For $r \leq t \leq s-1$, the cities of A_t are $\{1, 2, \ldots, t+1\} \setminus r$. Finally, for $t > s$, the cities of A_t are $\{1, 2, \ldots, t\}$. We have thus established the "only if" part of the lemma.

Suppose instead that for each j, the cities of A_j are $\{1, 2, \ldots, j+1\} \setminus k$ for some $k \in \{2, \ldots, j+1\}$. Let $S = \{r\}$: the cities of A_r are $\{1, 2, \ldots, r\}$, and let us denote the cities in S as $\{1, j_2 \ldots, j_k\}$. Then one can establish inductively that A can be created from *InsertMove*$(j_s + 1, j_{s+1})$ for all s such that $j_s + 1 \neq j_{s+1}$. □

The Weak Insertion Dynasearch Neighborhood N_{WIDS} for the TSP

1. $1 \in N^*_{WIDS}$

2. Suppose that $A \in N^*_{WIDS}$ and $\max(A) = i$. Then

- a. $A \star i+1 \in N^*_{WIDS}$.
- b. if $|A| = i$, then $A \star i+2 \in N^*_{WIDS}$.
- c. if $|A| = i-1$, then $A \star j \in N^*_{WIDS}$, where $j = \{1, \ldots, i\} \backslash A$.

3. $N_{WIDS} = \tau(N^*_{WIDS} \cap S_n)$.

```
State Space Graph $G_{WIDS} = (V_{WIDS}, E_{WIDS})$:
1. $(\{1\}, 1) \in V_{WIDS}$.
2. Suppose $(S, k) \in V_{WIDS}$ and $\max(S) = i$. Then $(S \cup \{j\}, j) \in V_{WIDS}$ where
       a. $(S \cup \{i + 1\}, i + 1) \in V_{WIDS}$ and there is an arc from $(S, k)$ to $(S \cup \{i + 1\}, i + 1)$
              with cost $c(k, i + 1)$.
       b. if $|S| = i$, then $(S \cup \{i + 2\}, i + 2) \in V_{WIDS}$ and there is an arc from $(S, k)$ to
              $(S \cup \{i + 2\}, i + 2)$ with cost $c(k, i + 2)$.
       c. if $|S| = i - 1$, then $(S \cup \{j\}, j) \in V_{WIDS}$ for $j = \{1, \ldots, i\} \setminus A$ and there is an arc from
              $(S, k)$ to $(S \cup \{j\}, j)$ with cost $c(k, j)$.
3. If $(S, j) \in V_{BS}$ and if $|S| = n$, then there is an arc from $(S, j)$ to $t$ with cost $c(j, 1)$.
```

Theorem 2. *The neighborhood N_{WIDS} is the weak insertion dynasearch neighborhood. The corresponding state space graph G_{WIDS} has $O(n^2)$ nodes and $O(n^2)$ edges. The time to find a minimum distance neighbor is $O(n^2)$.*

Proof. Lemma 3 establishes that N_{WIDS} is the weak insertion dynasearch neighborhood.

By Lemma 3, there are $O(n^2)$ values for (S, k) because $S = \{1, \ldots, j\}$ and $1 < k \le j$ or else $S = \{1, \ldots, k\} \setminus \{i\}, 1 < i < k$. In both cases, the number of arcs emanating from each state is equal to 2, and thus there are $O(n^2)$ edges. Moreover, the graph G_{WIDS} is acyclic and can be created in $O(n^2)$ steps. It follows that the time to find a shortest path in G_{WIDS} from node $(\{1\}, 1)$ to node t is $O(n^2)$. □

4. The dynamic programming expansion of a neighborhood

For a given tour A, let $V_{TSP}[A]$ be the set of states of V_{TSP} associated with A, as defined in Section 3, and for a given collection N of tours, we let $V_{TSP}[N] = \bigcup_{A \in N} V_{TSP}[A]$.

In Section 3, we associated a neighborhood $N_{TSP}[V]$ with a subset V of states. In this section, we consider the inverse operation of associating a set of states $V_{TSP}[N]$ with a neighborhood. Moreover, for a given neighborhood N, we can first associate the set of states $V' = V_{TSP}[N]$, and then create a second neighborhood $N_{TSP}[V']$. We refer to the neighborhood $N_{TSP}[V'] = N_{TSP}[V_{TSP}[N]]$ as the *expansion* of neighborhood N with respect to the dynamic program DP_{TSP}, or more briefly as the *dynamic programming expansion* of N. In general, if N is any polynomially sized neighborhood, then N is a subset of the dynamic programming expansion N' of N, and N' can be searched in polynomial time. The primary result of this section is that the dynamic programming expansion of the 2-exchange neighborhood is the dynasearch neighborhood obtained by compounding weakly independent 2-exchanges.

We first state four propositions about the operations $V_{TSP}[N]$ and $N_{TSP}[V]$, each of which has a very straightforward proof that is omitted. We give a proof of the fifth proposition.

Proposition 1. *For a given set $N \subseteq S_n$, $V_{TSP}[N]$ is the smallest subset $V \subseteq V_{TSP}$ of states such that $N \subseteq N_{TSP}[V]$.*

Proposition 2. *For a given subset $V' \subseteq V_{TSP}$ of states, $N_{TSP}[V']$ is the largest subset $N \subseteq \tau(S_n)$ such that $V_{TSP}[N] \subseteq V'$.*

Proposition 3. *For a given set $N \subseteq \tau(S_n)$, $N \subseteq N_{TSP}[V_{TSP}[N]]$.*

Proposition 4. *For a given subset $V \subseteq V_{TSP}$ of states, $V_{TSP}[N_{TSP}[V]] \subseteq V$.*

Proposition 5. *Suppose that N_n is a neighborhood whose size is polynomially bounded in n. Then one can find the best neighbor in $N_{TSP}[V_{TSP}[N]]$ in polynomial time.*

Proof. Since $|N_n| = O(p(n))$ for some polynomial $p(n)$, it follows that $|V_{TSP}[N]| = O(np(n))$, and the number of edges of the corresponding state space graph is $O(n^2 p(n))$. Thus, the dynamic program can be solved in $O(n^2 p(n))$ time. □

4.1. The 2-exchange neighborhood and its dynamic programming expansion

Let N_{EX} denote the 2-exchange neighborhood, which can be obtained from $\tau(\langle 1, \dots, n\rangle)$ by performing at most one operation $RevMove(i, j)$ for $1 < i < j \le n$. Recall that the neighborhood N_{WDS} is obtained from $\tau(\langle 1, \dots, n\rangle)$ by permitting the compounding of weakly independent 2-exchanges that exclude city 1. Thus any element of N_{WDS} can be obtained from $\tau(\langle 1, \dots, n\rangle)$ by performing a sequence of 0 or more operations $RevMove(i_1, i_2)$, $RevMove(i_3, i_4), \dots, RevMove(i_{2j-1}, i_{2j})$, where $1 < i_1 < i_2 < \cdots < i_{2j}$.

Theorem 3. *The neighborhood N_{WDS} obtained by compounding weakly independent 2-exchanges is the dynamic programming expansion of the 2-exchange neighborhood N_{EX}; that is, $N_{WDS} = N_{TSP}[V_{TSP}[N_{EX}]]$.*

Proof. We will first establish that $N_{WDS} \subseteq N_{TSP}[V_{TSP}[N_{EX}]]$, by proving the following claim.

Claim 1. $V_{TSP}[N_{EX}] = V_{TSP}[N_{WDS}]$.

From Claim 1, it will follow that $N_{TSP}[V_{TSP}[N_{WDS}]] = N_{TSP}[V_{TSP}[N_{EX}]]$, and by Proposition 3, $N_{WDS} \subseteq N_{TSP}[V_{TSP}[N_{WDS}]] = N_{TSP}[V_{TSP}[N_{EX}]]$.

Proof of Claim 1. It is clear that $V_{TSP}[N_{EX}] \subseteq V_{TSP}[N_{WDS}]$, and so it suffices to show that any state in $V_{TSP}[N_{WDS}]$ is also in $V_{TSP}[N_{EX}]$. Suppose that (S, k) is a state in $V_{TSP}[N_{WDS}]$. Then there is a sequence $A \in N_{WDS}$ such that A is obtained by carrying out a sequence of weakly independent reversal moves. Suppose $State(A_r) = (S, k)$ for some initial subsequence A_r of A. We need to prove that $(S, k) \in V_{TSP}(N_{EX})$. If city r is not part of any reversal, then $S = \{1, 2, \dots, r\}$ and $State(A_r) = (\{1, \dots, r\}, r) \in V_{TSP}(N_{EX})$ and the claim is true. So, we now consider the case that city r is part of a reversal, say $Rev[i, j]$. Let B be the sequence in the 2-exchange neighborhood obtained by performing only $RevMove[i, j]$, and let B_r consist of the first r cities of B in order. Then $State(B_r) = State(A_r)$, and so Claim 1 is true.

We will soon establish that $N_{TSP}[V_{TSP}[N_{EX}]] \subseteq N_{WDS}$; but first we state a characterization of $V_{TSP}[N_{EX}]$, followed by a property of N_{WDS}. Both are elementary and stated without proof.

Claim 2. $V_{TSP}[N_{EX}] = \{(S, j) : S = \{1, \dots, k\}$ *for some* $k \ge j\} \cup \{(S, j) : S = \{1, \dots, i\} \cup \{j, \dots, k\}$ *for some* $i, j,$ *and* k *with* $i + 2 \le j \le k\}$.

Claim 3. *Suppose that $A = \langle i_1, \dots i_n\rangle$. The tour $\tau(A)$ is in N_{WDS} if the following is true:*

1. *$i_1 = 1$ and*
2. *if $i_j > i_{j+1}$, then $i_{j+1} = i_j - 1$.*

Let $V = V_{TSP}[N_{EX}]$. We now prove that any tour $\tau(A) \in N_{TSP}[V]$ satisfies the properties of Claim 3, and is therefore in N_{WDS}. Let $A = \langle i_1, \dots, i_n\rangle$, and let $A_j = \langle i_1, \dots, i_j\rangle$ for each j. It is clear that $i_1 = 1$ because the only state in V with one element is $(\{1\}, 1)$. Now suppose that $i_j > i_{j+1}$. Then $State(A_j) = (S, i_j) \in V$, and $S = \{1, \dots, i'\} \cup \{i_j, \dots, k\}$ for some $i' < i_{j+1}$ and some $k \ge i_j$. Then $State(A_{j+1}) = \{1, \dots, i'\} \cup \{i_{j+1}, \dots, k\}$, because there is no other possible state of V that is consistent with A_{j+1} and thus $i_{j+1} = i_j - 1$, and A satisfies Claim 3, and is thus in N_{WDS}. □

Acknowledgements

Özlem Ergun was supported in part under NSF grant DMI-0238815. James B. Orlin was supported in part under ONR grant N00014-98-1-0317 and under NSF grant DMI-0217123.

References

[1] E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley & Sons, New York, 1997.
[2] R. Agarwal, Ö. Ergun, J.B. Orlin, C.N. Potts, Parallel machine scheduling problems with variable depth local search, Working paper, School of Ind. and Sys. Engr., Georgia Institute of Technology, Atlanta, 2003.
[3] R.K. Ahuja, Ö. Ergun, J.B. Orlin, A.B. Punnen, A survey of very large-scale neighborhood search techniques, Discrete Applied Mathematics 123 (2002) 75–102.

[4] E. Balas, New classes of efficiently solvable generalized salesman problems, MSRR No. 615, GSIA, Carnegie Mellon University, Pittsburgh, 1996.

[5] E. Balas, N. Simonetti, Linear time dynamic programming algorithms for new classes of Restricted TSPs, INFORMS Journal on Computing 13 (2001) 56–75.

[6] A. Bompadre, J.B. Orlin, Using grammars to generate very large scale neighborhoods for the traveling salesman problem and other sequencing problems, in: Proceedings of the 11th International IPCO Conference, Berlin, Germany, 2005, pp. 437–451.

[7] R.K. Congram, Polynomially searchable exponential neighborhoods for sequencing problems in combinatorial optimization, Ph.D. Dissertation, Faculty of Mathematical Studies, University of Southampton, UK, 2000.

[8] R.K. Congram, C.N. Potts, S.L. van de Velde, An iterated dynasearch algorithm for the single machine total weighted tardiness scheduling problem, INFORMS Journal on Computing 14 (2002) 52–67.

[9] V. Deĭneko, G.J. Woeginger, A study of exponential neighborhoods for the traveling salesman problem and the quadratic assignment problem, Mathematical Programming 87 (2000) 519–542.

[10] Ö. Ergun, New neighborhood search algorithms based on exponentially large neighborhoods, Ph.D. Dissertation, Operations Research Center, MIT, Cambridge, 2001.

[11] Ö. Ergun, J.B. Orlin, A. Steele-Feldman, Creating very large scale neighborhoods out of smaller ones by compounding moves, Journal of Heuristics (2002) (in press).

[12] G.M. Gutin, A. Yeo, A. Zverovitch, Exponential neighborhoods and domination analysis for the TSP, in: G. Gutin, A.P. Punnen (Eds.), The TSP and Its Variations, Kluwer Academic Publishers, Boston, 2002.

[13] M. Held, R.M. Karp, A dynamic programming approach to sequencing problems, Journal of SIAM 10 (1962) 196–210.

[14] C.N. Potts, S.L. van de Velde, Dynasearch—iterative local improvement by dynamic programming: part I, The traveling salesman problem, Faculty of Mathematical Studies, University of Southampton, UK, 1995 (preprint).