**Bab 8**

**Insertion-based Heuristik**

**8.2.2 Insertion-based heuristics**

Pendekatan nearest-neighbor memiliki banyak batasan yang jelas, yang telah kami sebutkan. Yang lain adalah fakta bahwa itu memungkinkan AS untuk menambahkan kota hanya pada *akhir* urutan saat ini. Kami dapat mengizinkan penyisipan di titik mana pun dalam urutan. Karena kita harus kembali ke titik awal, akan lebih baik untuk memperluas rute tertutup, daripada urutan terbuka yang kita tutup pada langkah terakhir dari prosedur. Gagasan ini mengarah pada insertion heuristics, yang masih sangat sederhana. Pada setiap langkah algoritma, kami memiliki satu set  dari kota-kota residual untuk dikunjungi dan tur sebagian ; yang kita butuhkan adalah memilih busur di , yang harus "dibuka" untuk memungkinkan penyisipan kota baru antara dan , yang mengarah ke urutan berikutnya . Sebenarnya, dengan sebagian rute, kita harus membuat dua keputusan:

1. Kota mana  untuk dimasukkan dalam ;
2. Titik penyisipan yaitu antara kota mana dan sudah ada di  kita harus memasukkan .

Karena kami menganggap properti triangularity berlaku, memasukkan kota baru hanya dapat meningkatkan panjang total dari rute parsial saat ini. Oleh karena itu, kriteria yang masuk akal adalah membuat keputusan sedemikian rupa untuk meminimalkan biaya tambahan penyisipan. Biaya tambahan untuk memasukkan kota antara dan adalah

Panjang tambahan ini biasanya disebut **jarak tempuh ekstra**.

Poin pertama yang harus kita perhatikan adalah bagaimana menemukan rute parsial awal. Salah satu kemungkinan adalah memilih busur terpendek dan biarkan menjadi rute parsial awal. Untuk menemukan kota berikutnya untuk disisipkan dalam rute parsial, kami dapat mencari  untuk kota terdekat dengan , yaitu, kami dapat menyelesaikan

Kemudian, mengingat kota baru (memutuskan hubungan secara sewenang-wenang), kami dapat mencari titik penyisipan terbaik dengan meminimalkan jarak tempuh tambahan. Prosedur ini diulangi sampai kita memiliki tur lengkap.

**Contoh 8.2** Mari kita perhatikan TSP gambar 8.2 lagi. Ada dua kota di rute awal. Memilih busur terpendek dalam jaringan, kami menetapkan rute awal sebagai . Kota terdekat dengan yang termasuk dalam  adalah kota 1. Untuk saat ini, tidak ada derajat kebebasan substansial dalam memilih titik penyisipan, dan kami memperbarui rute parsial . Rute ini setara dengan , karena masalahnya simetris dan cara kita melakukan perjalanan tur tidak relevan.

Sekarang kota terdekat dengan yang ada di  adalah kota 4, karena jaraknya dari kota 1 adalah 28, sedangkan jarak antara kota 2 dan 5 adalah 29. Sekarang kita harus menemukan titik penyisipan yang optimal di antara tiga kemungkinan berikut:

Oleh karena itu, kami menetapkan . Perhatikan bahwa tidak perlu mengevaluasi kembali seluruh tur setelah penyisipan, karena hanya biaya tambahan penyisipan yang diperlukan untuk membuat keputusan. Akhirnya, kita harus mengakomodasi kota 2:

Rute terakhir yang kami dapatkan adalah (3,5,2,4,1,3), dengan total panjang 124.

Dalam hal ini, kami mendapatkan solusi optimal, tetapi ini tidak dijamin secara umum, karena insertion-based heuristic masih merupakan greedy heuristic. Kami dapat mewakili prosedur konstruktif dasar kami sebagai cara greedy untuk menjelajahi pohon pencarian, sebuah konsep yang kami perkenalkan di bagian B.6.1 tentang branch and bound methods. Dalam branch and bound methods, kami memangkas cabang pohon pencarian hanya jika kami yakin itu tidak dapat menghasilkan solusi yang optimal. Dalam greedy heuristics, kita pada dasarnya memilih cabang yang paling menjanjikan, melupakan yang lain. Namun, kita dapat mengurangi perilaku myopic dari greedy heuristics dengan mengadopsi look-ahead strategy, di mana kita mengeksplorasi konsekuensi pilihan dengan memeriksa konsekuensinya beberapa langkah lebih jauh. Masalah selanjutnya menyangkut memutuskan hubungan ketika kita harus membuat keputusan. Dalam insertion-based procedures, kami mungkin memiliki dua titik penyisipan dengan jarak tempuh ekstra yang sama; dalam nearest-neighbor heuristic, kita mungkin memiliki dua kota atau lebih dengan jarak yang sama dari yang terakhir dalam urutan parsial. Dalam kasus seperti itu, kami dapat menjelajahi konsekuensi dari setiap alternatif sedikit lebih dalam di pohon pencarian, daripada memutuskan hubungan secara sewenang-wenang dan mengambil cabang yang pada dasarnya acak.

Dalam kasus khusus dari insertion-based approach di atas, kami juga dapat mencoba meningkatkan hasil, dengan biaya komputasi tambahan, dengan mempertimbangkan semua pasangan yang mungkin terdiri dari kota baru untuk disisipkan dan titik penyisipannya. Bahkan, dalam prosedur di atas kami memilih kota, dan kemudian kami menjelajahi kemungkinan titik penyisipan; kami dapat menemukan titik penyisipan yang optimal untuk setiap kota, dan hanya setelah evaluasi hasilnya kami membuat keputusan. Variasi lain pada tema adalah pilihan tur dua kota pertama; kita bisa mulai dari dua kota terjauh, bukan dari pasangan terdekat.