Algorithmica (1997) 18: 544–559

Algorithmica

A General Class of Heuristics for Minimum Weight Perfect Matching and Fast Special Cases with Doubly and Triply Logarithmic Errors[1]

C. Imielińska[2] and B. Kalantari[3]

Abstract. We give a class of heuristic algorithms for minimum weight perfect matching on a complete edge-weighted graph $K(V)$ satisfying the triangle inequality, where V is a set of an even number, n, of vertices. This class is a generalization of the Onethird heuristics, the hypergreedy heuristic, and it possibly employs any given exact or approximate perfect matching algorithm as an auxiliary heuristic to an appropriate subgraph of $K(V)$. In particular, by using the heuristic of Gabow *et al.* [3] as its auxiliary heuristic, our algorithm can obtain a solution whose weight is at most $(\frac{3}{2}\log_3\log_3\log_3 n + 2)$ times the weight of the optimal solution in $O(n^2 \log\log\log n)$ time, or a solution with an error of $3(\log_3\log_3 n)^{0.125} - 1$ in $O(n^2)$ time.

Key Words. Perfect matching, Heuristic algorithms.

1. Introduction.

We consider $K(V)$, a complete edge-weighted graph, satisfying the triangle inequality on a set of even number $n = |V|$ of vertices. A *perfect matching* of V is a set of edges such that each vertex of V is incident to exactly one edge. An *optimal perfect matching* of V is a perfect matching with minimum total edge weight. The optimal perfect matching can be obtained by Edmonds' algorithm [1], and its modifications by Gabow [2] and Lawler [7] in $O(n^3)$ time.

Because for large n the exact algorithms are not efficient enough, finding approximate solutions, fast and within some error bounds, has been of both practical and theoretical interest. By the *error* of a heuristic algorithm we mean the worst-case ratio of an approximate solution produced by the heuristic to that of the optimal solution.

We propose a class of heuristics for perfect matching, called the (t, k)-heuristic, where t and k are given integer parameters satisfying $0 \le t, k \le \lfloor \log_3 n \rfloor$. The (t, k)-heuristic is a generalization of the *hypergreedy* heuristic of Plaisted *et al.* [9], [10], and the *Onethird* class of heuristics by Grigoriadis and Kalantari [5]. The (t, k)-heuristic consists of $(r+1)$ stages, where $r \le k$. The first r stages are based on a combination of the above two heuristics. If $r < k$, the error of the (t, k)-heuristic is bounded above by $(2t+1)^{r+1} - 1$. If $r = k$, its $(r+1)$th stage makes use of any auxiliary exact or approximate perfect matching algorithm $\mathcal{A}$, and its error is bounded above by $(2t+1)^r[1 + f_{\mathcal{A}}(n_r)] - 1$, where $f_{\mathcal{A}}(n_r)$ is the error of the algorithm $\mathcal{A}$, applied to a complete graph with $n_r \le (1/3^t)^r n$ vertices. With appropriate choice of k the time complexity of the (t, k)-heuristic is $O(tn^2)$.

[1] This research was supported in part by the NSF under Grant No. CCR-9208371 and by the DIMACS Grant No. NSF-STC 88-09648.

[2] Department of Electrical Engineering and Computer Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA.

[3] Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA.

Received July 21, 1994; revised November 28, 1995. Communicated by C. H. Papadimitriou.

For $t = \lfloor \log_3 n \rfloor$, and $k = 1$, the (t, k)-heuristic reduces to the hypergreedy and runs in $O(n^2 \log n)$ time. If $t = 1$, and $0 \leq k \leq \lfloor \log_3 n \rfloor$, the (t, k)-heuristic becomes a greedy version of the Onethird class of heuristics, which runs in $O(\max\{n^2, t_{\mathcal{A}}(n_k)\})$ time, where $O(t_{\mathcal{A}}(n_k)$ is the time complexity of the auxiliary algorithm $\mathcal{A}$, possibly applied in the last stage of the heuristic. The class of (t, k)-heuristics generalize the Onethird heuristics and uses some properties of the hypergreedy heuristic. This combination results in a class of heuristics, which improves the error bounds of the corresponding Onethird heuristics.

Goemans and Williamson [4], have obtained a heuristic for perfect matching in complete graphs satisfying the triangle inequality which produces an approximate solution bounded above by twice the optimal weight, and runs in $O(n^2 \log n)$ time. Later, the running time was improved by Gabow, Goemans, and Williamson [3] (*GGW*) to $O(n^2\sqrt{\log\log n})$. Although the *GGW* algorithm has the interesting property of obtaining solutions to within the constant error of two, one might be interested in obtaining even faster heuristics with reasonably small theoretical error. Clearly, on the surface the *GGW* algorithm is superior to the hypergreedy and does not leave any incentive ever to use the latter. In fact using the *GGW* algorithm in the last stage of Onethird, already produces a better heuristic than the hypergreedy (see below). As we shall see the (t, k)-heuristic, which make use of the hypergreedy, is more powerful than the Onethird, and in conjunction with the *GGW* algorithm, as its auxiliary heuristic, results in fast heuristics with doubly and triply logarithmic errors.

More specifically, suppose that in the $(k + 1)$th stage we use the *GGW* algorithm. Onethird (or (t, k)-heuristic with $t = 1$) with $k = \frac{1}{4}\log_3\log_3\log_3 n$ gives a heuristic with $O(n^2)$ time complexity and $3(\log_3\log_3 n)^{0.25} - 1$ error. Even this heuristic is better than the hypergreedy, both in time complexity and error. In the (t, k)-heuristic with $t = 4$, $k = \frac{1}{16}\log_3\log_3\log_3 n$, we again get an $O(n^2)$-time heuristic whose error, $3(\log_3\log_3 n)^{0.125} - 1$, is even better than that of the Onethird. Finally, for $k = 1$, $t = \frac{1}{4}\log_3\log_3\log_3 n$, we obtain a heuristic whose error is $(\frac{3}{2}\log_3\log_3\log_3 n + 2)$. The corresponding time complexity is $O(n^2 \log\log\log n)$, still an improvement over the $O(n^2\sqrt{\log\log n})$ time of the *GGW* algorithm.

The (t, k)-heuristic makes use of a subgraph of $K(V)$, called the *t-basic graph*, denoted by $BG_t(V)$, which is a collection of sparse connected components, selected from $K(V)$. In Section 2 we describe the construction of the t-basic graph. In Section 3 we describe the (t, k)-heuristic. In Sections 4 and 5 we analyze the error and the time complexity of the (t, k)-heuristic, respectively. In Section 6 we analyze the (t, k)-heuristic with the *GGW* as its auxiliary algorithm.

2. The *t*-Basic Graph.

In this section we describe the construction of the t-basic graph and we analyze its time complexity. For $t = \lfloor \log_3 n \rfloor$, the t-basic graph was made use of in the hypergreedy heuristic [9]. Its time complexity was also analyzed in that paper. However, in this section we reanalyze the construction and the time complexity, in a more clear and simplified fashion than that of [9]. We construct the t-basic graph for any given $0 \leq t \leq \log_3 n$. Actually the 1-basic graph is simply the nearest-neighbor graph and it was used in the Onethird heuristic.

Given a subset W of the vertices V, the t-basic graph $BG_t(W)$, is a forest of trees, spanning W. The t-basic graph is constructed recursively from the $(t - 1)$-basic graph

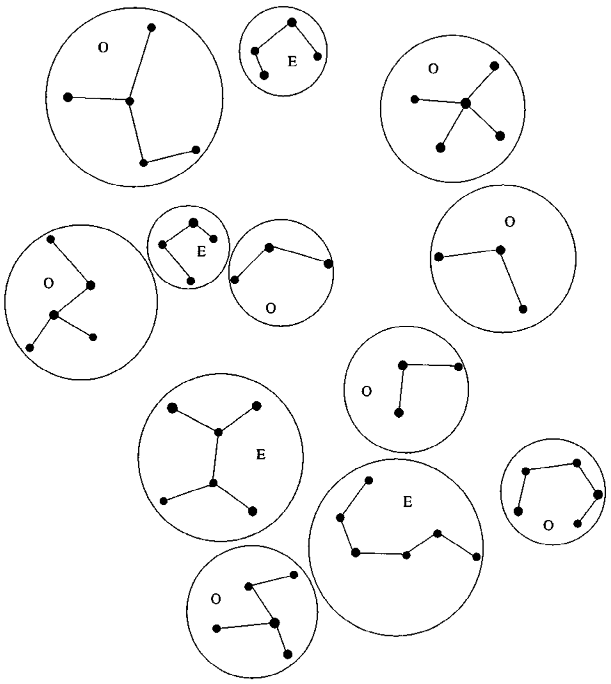

Fig. 1. The 1-basic graph—the nearest-neighbor graph of $K(W)$.

using edges in $K(W)$. The $(t-1)$-basic graph is a subgraph of the t-basic graph. The main feature of the t-basic graph is that its total weight is bounded above by a certain factor of the weight of the optimal perfect matching of $K(W)$. By a *partial matching* of W we mean a perfect matching of a subset of W. From the t-basic graph we extract a partial matching which will become a part of the final approximate solution.

The 1-basic graph, $BG_1(W)$, is the first graph formed and it is the nearest-neighbor graph of $K(W)$ (Figure 1). Let $O_1(W)$ and $E_1(W)$ be the *odd* and the *even* connected components of $BG_1(V)$, with odd and even number of vertices, respectively. We refer to these components as *hypervertices*.

To get $BG_2(W)$ from $BG_1(W)$, for each odd hypervertex in $O_1(W)$ we find its nearest odd hypervertex, where the two are connected either by an edge or a set of edges forming a shortest path in $K(W)$, possibly passing through the even hypervertices of $E_1(W)$ (Figure 2). The nearest-neighbor graph of the odd hypervertices contains old and new even components, $E_2(W)$, and new odd components, $O_2(W)$, which all together form $BG_2(W)$ (Figure 3).

This recursive procedure is repeated until the t-basic graph is formed. In general, given $BG_{i-1}(W)$ we obtain $BG_i(W)$ by adding a set of edges in $K(W)$ whose total weight is bounded above by twice the weight of the optimal perfect matching of $K(W)$. We denote by M_W^* and $\omega(M_W^*)$ an optimal perfect matching of $K(W)$ and its total edge weight, respectively.

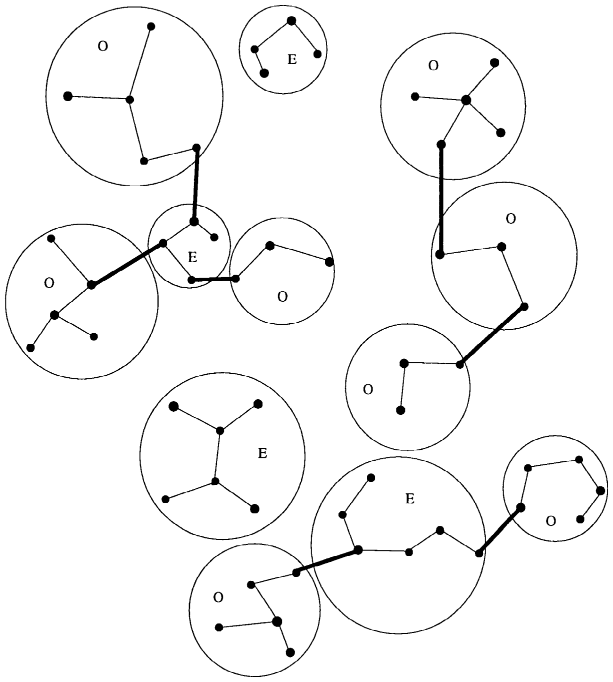

Fig. 2. The nearest-neighbor graph of odd hypervertices.

LEMMA 2.1. *For each* $1 \leq i \leq t$ *the total weight of all the edges added to* $BG_i(W)$ *to form a nearest neighbor graph of odd hypervertices is bounded above by* $2\omega(M_W^*)$.

PROOF. We consider, for a given i, the union of $BG_i(W)$ and an optimal perfect matching of $K(W)$, M_W^*. Edges of M_W^* partition the set of odd hypervertices into pairs, which are connected by either an edge in M_W^* or a chain of edges in M_W^* passing through even components of $E_i(W)$.

In Figure 4 we show one such pair of odd hypervertices, A and B. Let A_1 and B_1 be the nearest odd hypervertices of A and B, respectively. The weight of the shortest path connecting A and A_1 is less than or equal to the weight of the shortest path connecting A to B. Similarly the weight of the shortest path connecting B and B_1 is less than or equal to the weight of the shortest path connecting A to B. Thus the total weight of the nearest-neighbor graph of all the odd hypervertices of $BG_i(W)$ is bounded above by twice the weight of M_W^*. From Lemma 2.1 we set,

THEOREM 2.1. *The total weight of the* t*-basic graph* $BG_t(W)$ *of* $K(W)$ *is bounded above by* $2t\omega(M_W^*)$.

In particular, when $t = \lfloor \log_3 |W| - 1 \rfloor$, the t-basic graph is a collection of only even hypervertices, and if additionally $W = V$, this corresponds to the original hypergreedy heuristic [10]. In the hypergreedy, the even hypervertices, where each of them admits a

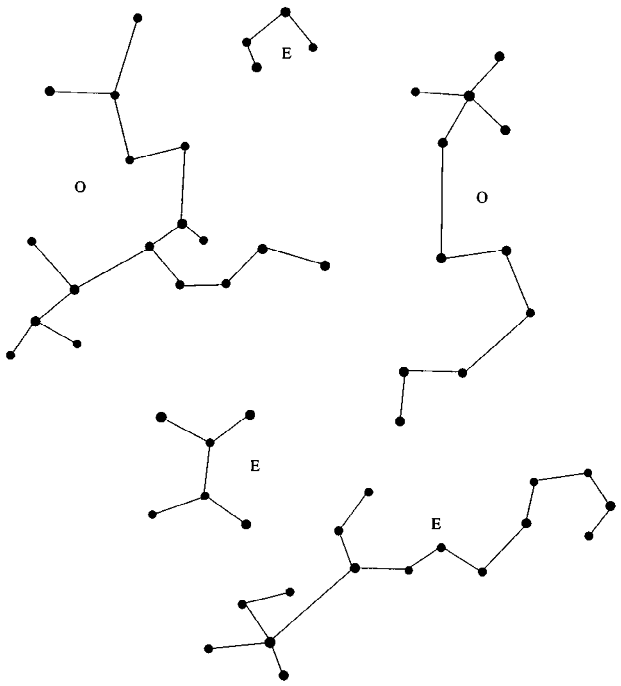

Fig. 3. The 2-basic graph with its odd and even hypervertices.

perfect matching, are converted into a perfect matching of the input graph, and the total weight of the matching does not exceed the weight of the t-basic graph.

2.1. *Time Complexity.* Here we analyze the time complexity for constructing the t-basic graph. The t-basic graph is constructed in t steps. The 1-basic graph, i.e., the nearest neighbor graph of W, is formed in $O(|W|^2)$ time. The time complexity of one

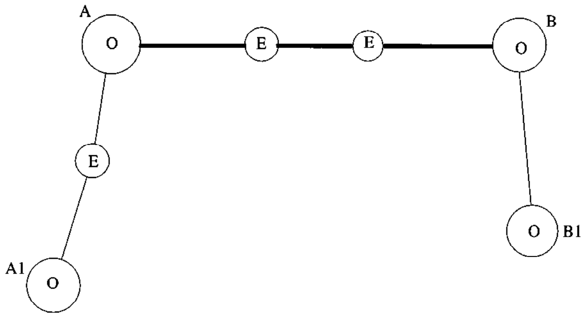

Fig. 4. A pair of odd hypervertices and their nearest odd neighbors; ——, edge in the nearest-neighbor graph, ▬▬, edge in the optimal perfect matching.

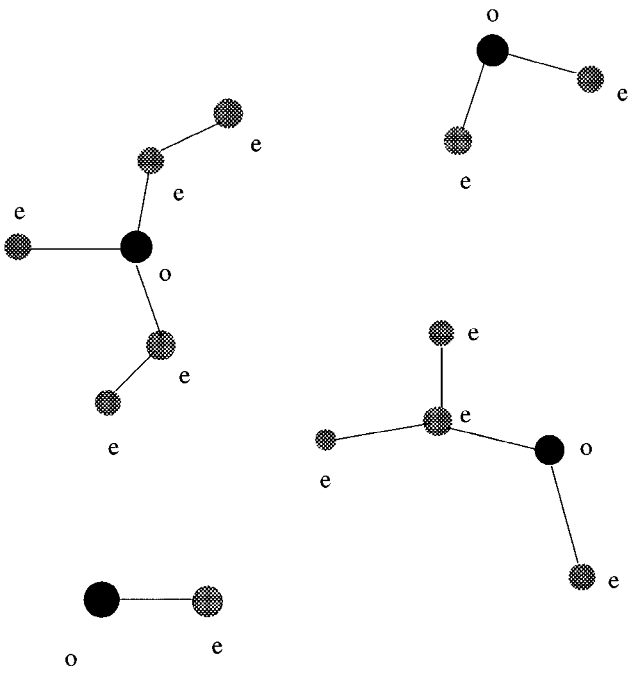

Fig. 5. Generalized Voronoi Diagram.

step of the recursive procedure, when the i-basic graph is obtained from the $(i-1)$-basic graph $(1 < i \leq t)$, is the time complexity for the construction of the nearest-neighbor graph of the odd hypervertices.

The construction of the nearest neighbor graph, consists of two stages. First, we build the *Generalized Voronoi Diagram* (*GVD*) *relative to the set of odd hypervertices*, which is a partition of all hypervertices with respect to which odd hypervertex they are closest to (Figure 5). Then we find for each odd hypervertex its nearest odd neighbor. In a GVD, for every even hypervertex a shortest path to its nearest odd hypervertex (possibly through other even hypervertices) is found. Each odd hypervertex and all the even hypervertices, which are closer to it than to any other odd hypervertex, form a *Generalized Voronoi Region* (*GVR*). Every even hypervertex is connected to its nearest odd hypervertex by a path constructed from the edges in $K(W)$. We say that two GVRs are *adjacent* if there is an edge in $K(W)$ with endpoints in the corresponding GVRs. Since there is always such an edge, each two GVRs are adjacent.

Before we analyze the time complexity of the construction of the t-basic graph, we consider the problem of computing the nearest-neighbor graph of the odd hypervertices in $BG_i(W)$, $1 \leq i \leq t-1$.

THEOREM 2.2. *Computing the nearest-neighbor graph of the odd hypervertices in* $BG_i(W)$ *can be done in* $O(|W|^2)$ *time.*

The proof of Theorem 2.2 follows from the following two lemmas.

LEMMA 2.2. *The time complexity of constructing the GVD of $BG_t(W)$ is $O(|W|^2)$.*

PROOF. First, we define an auxiliary graph with a root node R to be connected to each odd hypervertex by a zero-weight edge. Let E and O be the sets of the even and odd hypervertices, respectively. We apply Dijkstra's shortest-path algorithm to find all shortest paths from the hypervertices to the root R. Consider an even hypervertex $e_1 \in E$ and the first odd hypervertex $o_1 \in O$ which appears on the shortest path from e_1 to R. We claim that o_1 is the nearest odd neighbor of e_1. We assume that $o_2 \neq o_1$ is the nearest odd neighbor of e_1. Then the weight of the path from e_1 to the root via o_2 would be smaller than the weight of the corresponding path from e_1 to the root via o_1, a contradiction. Therefore, the time complexity of constructing the GVD of $BG_t(W)$ is $O(|W|^2)$. $\square$

The following was proved in [9].

LEMMA 2.3. *Given an odd hypervertex y and GVR(y), the GVR containing y, assume z is a nearest odd neighbor of y. Then there exists an edge (u, v), such that shortest path from y to z consists of the path from y to u, $u \in$ GVR(y), the edge (u, v), $v \in$ GVR(z), and the path from v to z.*

By the above lemma, given the GVD, we can determine in $O(|W|^2)$ operations the nearest odd neighbor for each odd hypervertex in $BG_i(W)$, $1 \leq i \leq t-1$, using edges in $K(W)$. We examine all the edges in $K(W)$ and regard those with endpoints in different GVRs, and select for each pair of GVRs such edge, which minimizes the lengths of the shortest path between the corresponding odd hypervertices. Thus,

THEOREM 2.3. *The time complexity of constructing $BG_t(W)$ is $O(t|W|^2)$.*

3. Description of the (t, k)-Heuristic. The (t, k)-heuristic is a generalization of the Onethird heuristic for perfect matching, and the hypergreedy. The (t, k)-heuristic is defined by two given integer parameters, t and k ranging from 1 to $\lfloor \log_3 n \rfloor$. It consists of $(r+1)$ stages, $r \leq k$, where at each stage $j = 0, 1, \ldots, r-1$, a complete graph $K(V_j)$, $V_j \subseteq V$, is processed. Given $K(V_j)$, $V_0 = V$, we extract the t-basic graph, $BG_t(V_j)$. By using the t-basic graph, we select a partial matching which covers a portion or possibly all the vertices in V_j. We remove from V_j all the vertices matched by the partial matching, while all the remaining unmatched vertices form a complete graph $K(V_{j+1})$, to be processed in the next stage. After r stages, if $r = k$, the remaining unmatched vertices, V_{r+1}, are matched by any auxiliary perfect matching algorithm $\mathcal{A}$, which can be either another heuristic for perfect matching, or an exact algorithm.

For $t = 1$, the (t, k)-heuristic uses only the nearest-neighbor graph (1-basic graph), and reduces to the greedy variation of the Onethird heuristic which in turn allows the selection of a partial matching with at least $\lceil \frac{1}{3} n_j \rceil$ edges, $n_j = |V_j|$. The crucial property of the (t, k)-heuristic is that by using the t-basic graph we can select a partial matching, with at least $\lceil \frac{1}{2}((3^t - 1)/3^t)n_j \rceil$ edges. This is because the size of each of the odd hypervertices (if there are any) in the t-basic graph is at least 3^t. Thus at each stage, the larger the parameter t, the more edges can be selected into a partial matching and this results in a stronger class of heuristics than the Onethird. The appropriate choice of the parameters t and k depend on the specific auxiliary algorithm $\mathcal{A}$, to be used in the last stage. More formally,

```
($t, k$)-heuristic for general weights satisfying the triangle inequality

Input: $K(V)$, $V_0 = V$, $(0 \le t, k \le \lfloor \log_3 n \rfloor)$ and auxiliary heuristic $\mathcal{A}$.
Output: A perfect matching of $V$.

For each stage $j = 0, 1, \ldots, k-1$, if $V_j \neq \emptyset$: do
begin
1. Construct the $t$-basic graph, $BG_t(V_j)$.
2. In every connected component of $BG_t(V_j)$:
     (a) Duplicate edges.
     (b) Extract an Euler tour.
     (c) Convert the tour into a Hamiltonian cycle of lesser weight.
     (d) Find maximum cardinality minimum weight perfect matching in
         the cycle.
3. Select a matching formed by the union of the maximum cardinality mini-
   mum weight matchings obtained in Step 2(d), which is either a partial or
   a perfect matching of $V_j$.
4. All unmatched vertices, if there are any, form $K(V_{j+1})$.
end
Auxiliary Stage $(k+1)$: Match all the remaining vertices using an auxiliary
algorithm $\mathcal{A}$ for perfect matching.
```

For each j, we denote by ET_j, H_j, S_j, the union of Euler tours, Hamiltonian cycles, and maximum cardinality minimum weight perfect matchings obtained from all the connected component of $BG_t(V_j)$.

For $t = 1$ the above algorithm is equivalent to a greedy variation of the Onethird. However, the selection of the partial matching S_j in Step 3 is different in the original Onethird, where we first sort all the edges in H_j, the union of all Hamiltonians, then instead of Step 2(d) we select exactly $\frac{1}{2}\lfloor (3^t - 1)/3^t \rfloor$ shortest edges in such a way that one edge is chosen at a time and all the edges incident to it are removed.

For $j = 0, 1, \ldots, k-1$, let M_j^* be an optimal perfect matching of $K(V_j)$. At each stage j, $j = 0, 1, \ldots, k-1$, we construct the t-basic graph $BG_t(V_j)$ (Step 1), whose total weight does not exceed $2t\omega(M_j^*)$. There are two types of hypervertices in the t-basic graph, even and odd, with an even and odd number of vertices, respectively. Each even hypervertex admits a perfect matching.

To select a partial matching S_j of $K(V_j)$ from the t-basic graph, we first duplicate the edges in $BG_t(V_j)$ (Step 2(a)), which results in a new graph, where each component is Eulerian. A graph is said to be Eulerian if each of its vertices has an even degree. Such a graph admits a closed tour, called an Euler tour, which visits each edge exactly once, e.g., see [8]. We construct an Euler tour in every connected component of the t-basic graph

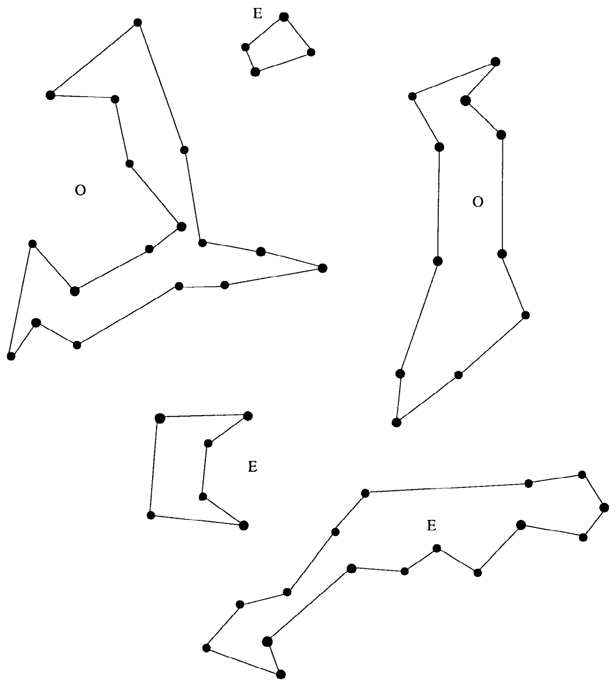

Fig. 6. Hamiltonian cycles, H_j, extracted from Euler tours, ET_j.

(Step 2(b)), and ET_j is the union of all such tours. Figure 3 can be viewed as the duplicated 2-basic graph. The weight of the edges in ET_j is bounded above by twice the weight of the t-basic graph. The Euler tours are converted into Hamiltonian cycles (Step 2(c) and Figure 6), where H_j is their union. A Hamiltonian cycle of a connected component is a cycle where each vertex is visited exactly once. From the triangle inequality, the total weight of the edges in H_j is less than or equal to the total weight of the edges in ET_j.

We find a maximum cardinality minimum weight matching in each Hamiltonian cycle (Step 2(d) and Figure 7). The union of such matchings contains at most half of all the edges in H_j, and the total edge weight of the union is bounded above by $\frac{1}{2}\omega(H_j)$. These edges (Figure 8) form a partial matching S_j. The total weight of S_j does not exceed half the weight of the Hamiltonians. The remaining unmatched vertices, if there are any, result in a new complete graph $K(V_{j+1})$, to be processed in the $(j+1)$th stage (Figure 8). We repeat this process until either we obtain a perfect matching of V in $r \le k$ stages or at the completion of kth stage we apply a perfect matching algorithm $\mathcal{A}$.

Each jth stage of the heuristic can be implemented in $O(t(n_j)^2)$ time. We show that if the parameter k is selected appropriately, the (t, k)-heuristic runs in $O(tn^2)$ time.

4. Analysis of the (t, k)-Heuristic.

In this section we present the analysis of the error of the (t, k)-heuristic. Given a subset W of V, let S be a partial matching selected from W and let M_W^* and $M_{W'}^*$ be the optimal perfect matchings of W and W', respectively,

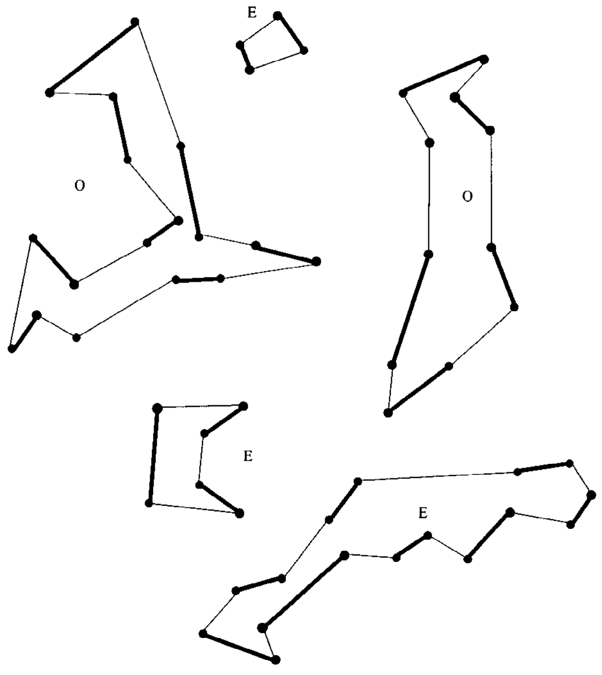

Fig. 7. The union of maximum cardinality minimum weight matchings, represented by the thick edges, obtained from H_j.

where W' is the set of unmatched vertices of W left after selecting S. The following relates the weights of $M^*_{W'}$, M^*_W, and S.

LEMMA 4.1 [5]. $\omega(M^*_{W'}) \leq \omega(M^*_W) + \omega(S)$.

LEMMA 4.2. *At each stage $j = 0, 1, \ldots, r-1$, $r \leq k$ of the (t, k)-heuristic we can always select from $K(V_j)$ a partial matching S_j containing $\lceil \frac{1}{2}((3^t - 1)/3^t)n_j \rceil$ edges.*

PROOF. First we claim that if there is an odd hypervertex in the t-basic graph, its size is at least 3^t. Note, if there is no odd hypervertex in the t-basic graph, then S_j is simply a perfect matching of V_j and the lemma is true. Clearly, this is true for $t = 1$. For $t > 1$, we only need to observe that an odd hypervertex in the t-basic graph is created from at least three odd hypervertices in the $(t-1)$-basic graph. Let $C_1, C_2, \ldots, C_l$ be the odd cycles, and let $C_{l+1}, \ldots, C_p$ be the even cycles in H_j. There are $(|C_i| - 1)/2 \geq (3^t - 1)/2$, $i = 1, \ldots, l$, and $|C_i|/2$, $i = l+1, \ldots, p$, edges in any maximum cardinality matching of each odd and even cycle, respectively. Thus we can select from each cycle at least $\lceil ((3^t - 1)/2)(|C_i|/3^t) \rceil$, $i = 1, \ldots, p$, vertex disjoint edges. The proof of the lemma is

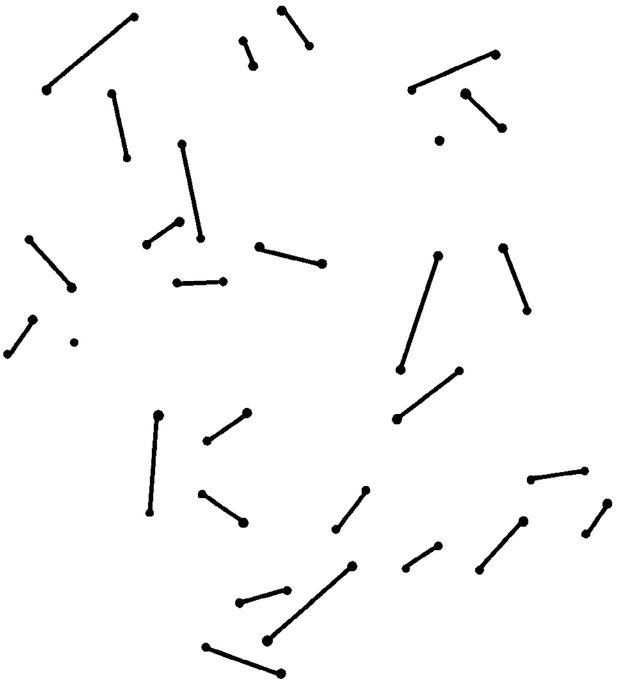

Fig. 8. Partial matching S_j formed the maximum cardinality minimum weight matchings of the Hamiltonian cycles ($t = 2$), and the remaining unmatched vertices to be processed in the next stage.

now immediate from the following inequalities:

$$\left\lceil \frac{1}{2}\left(\frac{3^t-1}{3^t}\right)n_j \right\rceil \le \sum_{i=1}^{p} \left\lceil \frac{3^t-1}{2}\frac{|C_i|}{3^t} \right\rceil. \qquad \square$$

LEMMA 4.3. *The partial matching S_j selected at each stage j, $j = 0, 1, \ldots, r-1$, $r \le k$, of the (t, k)-heuristic has weight bounded above by*

$$\omega(S_j) \le 2t\omega(M_j^*).$$

PROOF. The partial matching S_j is selected from H_j. The total weight of H_j is bounded above by $4t\omega(M_j^*)$, a bound on the duplicated weight of the t-basic graph of $K(V_j)$. We find a maximum cardinality minimum weight matching in each Hamiltonian cycle. If the size of each cycle is even, then, clearly, $\omega(S_j) \le \frac{1}{2}\omega(H_j)$, so that the lemma is proved. If the size of a cycle is odd, then using the triangle inequality it is easy to argue that the maximum cardinality minimum weight matching of that cycle must have weight less than half of that of the cycle. $\square$

A perfect matching of $K(V)$, produced by the (t, k)-heuristic is the union of partial matchings, S_j, selected at stages $j = 0, 1, \ldots, r-1$, $r \le k$. Let M_j be the perfect matching of $K(V_j)$, produced by the heuristic. Thus M_j is the union of partial matchings produced at stages $j, j+1, \ldots, k$, i.e., $M_j = \bigcup_{i=j}^{r} S_i$. In particular, M_0 is a perfect matching of $K(V)$ produced by the heuristic.

The error for stage j is the ratio of the weight of M_j to the weight of M_j^*, the optimal perfect matching of $K(V_j)$. We denote the error by $f(n_j) = \omega(M_j)/\omega(M_j^*)$, for $\omega(M_j^*) \neq 0$. For a degenerate case, when $\omega(M_j^*) = 0$, V_j can be viewed as a set of double points, and the distance between each two such points is 0, and the perfect matching M_j, produced by the heuristic is identical to M_j^*, therefore $f(n_j)$ can be defined to be 1. In the following lemma, we relate errors in two consecutive stages.

LEMMA 4.4. *For each* $j = 0, 1, \ldots, r-1, r \leq k$, *we have*

$$f(n_j) \leq 2t + [2t+1] f(n_{j+1}).$$

PROOF. From Lemma 4.1 we get $\omega(M_{j+1}^*) \leq \omega(M_j^*) + \omega(S_j)$. From this and Lemma 4.3 we get

$$\frac{\omega(M_{j+1}^*)}{\omega(M_j^*)} \leq 1 + 2t.$$

Given M_j and M_{j+1}, the perfect matching produced by the heuristic for $K(V_j)$ and $K(V_{j+1})$, respectively, we have

$$\omega(M_j) = \omega(S_j) + \omega(M_{j+1}).$$

From Lemma 4.3 we get

$$\omega(M_j) \leq 2t\omega(M_j^*) + \omega(M_{j+1}).$$

The proof of the lemma follows by dividing the above inequality by $\omega(M_j^*)$, writing

$$\frac{\omega(M_{j+1})}{\omega(M_j^*)} = \frac{\omega(M_{j+1})}{\omega(M_{j+1}^*)} \cdot \frac{\omega(M_{j+1}^*)}{\omega(M_j^*)},$$

and using the bound of the last ratio. □

The overall error of the (t, k)-heuristic can be expressed by the ratio

$$f(n) = f(n_0) = \frac{\omega(M_0)}{\omega(M_0^*)},$$

where M_0 is a perfect matching of $K(V)$, produced by the (t, k)-heuristic, and M_0^* is the optimal perfect matching of $K(V)$.

LEMMA 4.5. *For* $j = 0, 1, \ldots, r$ *we have*

$$n_j \leq \left(\frac{1}{3^t}\right)^j n.$$

PROOF. There are at least $\lceil((3^t-1)/3^t)n_j)\rceil$ vertices matched by S_j at stage j and the remaining n_{j+1} vertices which will be processed in the next stage satisfy

$$n_{j+1} \le n_j - 2\left\lceil \frac{1}{2}\left(\frac{3^t-1}{3^t}\right)n_j \right\rceil.$$

Note that we can write $((3^t-1)/2)n_j = 3^t m - r$, where m and $0 \le r \le 3^t - 1$ are integers. Thus

$$n_{j+1} \le n_j - 2\left\lceil \frac{3^t m}{3^t} - \frac{r}{3^t} \right\rceil = n_j - 2m.$$

Since $m = \frac{1}{2}((3^t-1)/3^t)n_j + r/3^t$, we get

$$n_{j+1} \le n_j - 2m = n_j - 2\frac{1}{2}\frac{3^t-1}{3^t}n_j - 2\frac{r}{3^t} = \frac{n_j}{3^t} - 2\frac{r}{3^t} \le \frac{n_j}{3^t}.$$

From the above the proof of the lemma is immediate. $\square$

THEOREM 4.1. *Let $K(V)$ be a complete edge-weighted graph with n vertices, satisfying the triangle inequality, the error of (t,k)-heuristic is bounded above by*

$$f(n) \le \begin{cases} (2t+1)^{r+1} - 1 & \text{for} \quad r < k, \\ (2t+1)^r[1 + f_{\mathcal{A}}(n_r)] - 1 & \text{for} \quad r = k, \end{cases}$$

where $n_r \le (1/3^t)^r n$, and $f_{\mathcal{A}}(n_r)$ is the error of the auxiliary algorithm $\mathcal{A}$.

PROOF. After r stages the (t,k)-heuristic either matches all the vertices in $K(V)$, or there are n_r unmatched vertices left. We apply recursively Lemma 4.4, and get the overall error $f(n) = f(n_0)$,

$$f(n) \le 2t\sum_{i=0}^{r-1}(2t+1)^i + (2t+1)^r z,$$

where $z = 2t$ if $r < k$, and $z = f_{\mathcal{A}}(n_r)$ if $r = k$. Equivalently,

$$f(n) \le (2t+1)^r[1+z] - 1. \qquad \square$$

5. Time Complexity of the (t,k)-Heuristic.

In this section we derive a bound on the worst-case time complexity of the (t,k)-heuristic.

THEOREM 5.1. *The (t,k)-heuristic can be implemented in $T(n) = O(tn^2)$ time.*

PROOF. For $j = 0, \dots, r-1$, from Theorem 2.3, the t-basic graph of $K(V_j)$ can be implemented in $O(t(n_j)^2)$ time. From the t-basic graph, we construct a collection of Euler tours, then Hamiltonian cycles, in $O(n_j)$ time. We claim that finding a maximum cardinality minimum weight matching of a Hamiltonian cycle is linear in the size of the

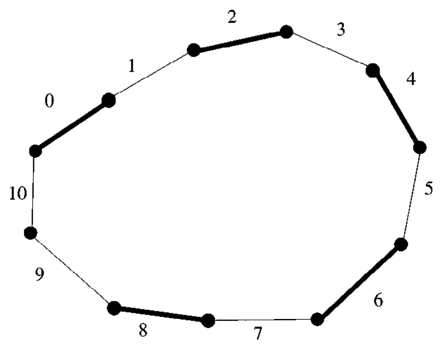

Fig. 9. One of eleven maximum cardinality matchings of a cycle of size 11.

cycle. Let the sequence $\{0, 1, \ldots, q-1\}$ correspond to the edges in a cycle of size q, in the order they appear. If q is even, then there are two different maximum cardinality matchings, each of size $q/2$, and we compute the weight of each of the two and select the one with the smaller weight. For q odd, there are q different matchings, each of size $(q-1)/2$. We can view the cycle as a circular list of q edges and each of its maximum cardinality matching as a list of $(q-1)/2$ edges, see Figure 9. First we generate the maximum cardinality matching corresponding to $\{0, 2, 4, \ldots, q-3\}$. We compute the weight of this matching in $O(q)$ time. We replace the first (the edge 0) with the last edge in the cycle (the edge $(q-1)$), and compute in a constant time the weight of the resulting new matching. We repeat this process $O(q)$ times and obtain the weight of all the maximum cardinality matchings from which we select the one with the smallest weight. Hence the proof of the above claim.

The union of the maximum cardinality minimum weight matchings of all the Hamiltonians forms a partial matching S_j. Therefore one stage of the (t, k)-heuristic can be implemented in $O(t(n_j)^2)$ time, and, from Lemma 4.5, the overall time complexity is bounded above by

$$\begin{aligned} T(n) &= O\left(\sum_{j=0}^{r-1} t(n_j)^2\right) = O\left(\sum_{j=0}^{r-1} t\left(\frac{1}{3^t}\right)^{2j} n^2\right) = O\left(tn^2 \sum_{j=0}^{r-1}\left(\frac{1}{3^t}\right)^{2j}\right) \\ &= O\left(tn^2 \frac{1}{1-1/3^{2t}}\right) = O(tn^2). \end{aligned} \qquad \square$$

6. Special Cases with Doubly and Triply Logarithmic Errors.

As a corollary of Theorem 4.1 and Theorem 5.1, we show that if we use the *GGW* algorithm as the auxiliary heuristic of the (t, k)-heuristic, the error of the resulting heuristic is a very slowly growing function of n and its time complexity is $O(tn^2)$. Thus we obtain heuristics which are faster than the *GGW* algorithm and have competitive errors.

COROLLARY 6.1. *The error of the (t, k)-heuristic for $t = 1, 2, \ldots, \frac{1}{4}\log_3\log_3\log_3 n$ and after $k = (1/4t)\log_3\log_3\log_3 n$, using the GGW algorithm as the possible auxiliary* *heuristic is bounded above by*

$$f(n) \leq 3(\log_3 \log_3 n)^{(1/4t)\log_3(1+2t)} - 1,$$

and its time complexity is $O(tn^2)$.

PROOF. Let $T_{\mathcal{A}}(n)$ be the time complexity of the auxiliary perfect matching algorithm $\mathcal{A}$. We want the parameter k to satisfy $O(T_{\mathcal{A}}(n_k)) = O(tn^2)$, which would guarantee that the overall time complexity of the resulting heuristic to remain would be $O(tn^2)$ time. For *GGW* $T_{\mathcal{A}}(n) = O(n^2\sqrt{\log\log n})$, and $f_{\mathcal{A}}(n) \leq 2$. If $r = k$ we use this algorithm to match the remaining n_k vertices. In order to find the appropriate choice for the parameters k and t, the following has to be satisfied:

$$(n_k)^2\sqrt{\log_3 \log_3} n_k \leq tn^2.$$

Since $n_k \leq (1/3^t)^k n$ (see Lemma 4.5), it is easy to check that the above is satisfied when $k = (1/4t)\log_3\log_3\log_3 n$ and $1 \leq t \leq \frac{1}{4}\log_3\log_3\log_3 n$. Thus from Theorem 4.1 we obtain

$$\begin{aligned} f(n) &\leq [2t+1]^k[1+f_{\mathcal{A}}(n_k)] - 1 \leq 3[2t+1]^{(1/4t)\log_3\log_3\log_3 n} - 1 \\ &= 3(\log_3\log_3 n)^{(1/4t)\log_3(1+2t)} - 1. \end{aligned}$$ □

From Corollary 6.1, the (t, k)-heuristic, for $t = 1$ and $k = \frac{1}{4}\log_3\log_3\log_3 n$ (which becomes a version of the Onethird heuristic), gives an $O(n^2)$-time heuristic with the error of $3(\log_3\log_3 n)^{0.25} - 1$. This heuristic is better than the hypergreedy both in time and error. For $t = 4$ and $k = \frac{1}{16}\log_3\log_3\log_3 n$, the (t, k)-heuristic also gives an $O(n^2)$-time heuristic with error bounded above by $3(\log_3\log_3 n)^{0.125} - 1$, which is even better than that of the Onethird. Finally, for $k = 1$ and $t = \frac{1}{4}\log_3\log_3\log_3 n$, we obtain a solution bounded above by $(\frac{3}{2}\log_3\log_3\log_3 n + 2)$. The corresponding time complexity is $O(tn^2) = O(n^2\log\log\log n)$, still an improvement over the $O(n^2\sqrt{\log_3\log_3})$ time of the *GGW* algorithm.

REMARKS. Our algorithm is an attractive alternative to the *GGW* algorithm for two reasons. First, it offers a *full range of speed-approximation tradeoffs*, whereas *GGW* has fixed performance. Second, in certain regions of this tradeoff it is faster than *GGW* (admittedly, only in an asymptotic sense); in other words, it relates to *GGW* as it relates to the exact algorithm: It sacrifices optimality for efficiency. However, it is worth noting in closing that, from the practical point of view, and for the important *Euclidean* case of the matching problem, even the *weak greedy* heuristic [6], whose theoretical error is $2^{n/2} - 1$, ironically provides experimental solutions with errors bounded within [1.29, 1.45].

Acknowledgments. We would like to thank an anonymous referee for simplifying the proof of Lemma 2.2, and Christos Papadimitriou for important suggestions.

References

[1] J. Edmonds, Paths, trees and flowers, *Canadian Journal of Mathematics*, 17 (1965), 449–467.

[2] H. N. Gabow, Implementation of algorithms on nonbipartite graphs, Ph.D thesis, Department of Electrical Engineering, Stanford University, 1973.

[3] H. N. Gabow, M. X. Goemans, and D. P. Williamson, An efficient approximate algorithm for the survivable network design problems, *Proc. of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, 1993, pp. 57–74.

[4] M. X. Goemans and D. P. Williamson, A general approximation technique for constrained forest problems, *Proc. of the Third Annual ACM–SIAM Symposium on Discrete Algorithms*, 1992.

[5] M. D. Grigoriadis and B. Kalantari, A new class of heuristic algorithms for weighted perfect matching, *Journal of the Association for Computing Machinery*, 35 (1988), 769–776.

[6] M. D. Grigoriadis, B. Kalantari and C. Y. Lai, On existence of weak greedy matching heuristics, *Operations Research Letters*, 5(4) (1986), 201–205.

[7] E. L. Lawler, *Combinatorial Optimization*: *Networks and Matroids*, Rinehart and Winston, New York, 1976.

[8] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*: *Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[9] D. A. Plaisted, Heuristic matching for graphs satisfying the triangle inequality, *Journal of Algorithms*, 5 (1984), 163–179.

[10] D. A. Plaisted, E. M. Reingold, and K. J. Supowit, Heuristic for weighted matching, *Proc. of the Symposium on the Theory of Computing*, 1980, pp. 398–419.