Part I

COMPUTER SCIENCE

BIT 33 (1993), 178–189.

A GENERALIZED HYPERGREEDY ALGORITHM FOR WEIGHTED PERFECT MATCHING

CELINA IMIELINSKA[1] and BAHMAN KALANTARI[2]

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

Department of Computer Science
Rutger University
New Brunswick, NJ 08903, USA

Abstract.

We give a generalization of the hypergreedy algorithm for minimum weight perfect matching on a complete edge weighted graph whose weights satisfy the triangle inequality. With a modified version of this algorithm we obtain a $\log n$-approximate perfect matching heuristic for points in the Euclidean plane, in $O(n \log^2 n)$ time.

CR categories: F.2.2, G.2.2, E.5.

Keywords: Perfect matching, Approximation algorithms.

1. Introduction.

Let $K(V)$ be a complete edge weighted graph with an even number, $n = |V|$, of vertices. Throughout the paper we shall assume that the edge weights satisfy the triangle inequality. *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 [4, 5], in $O(n^3)$ time for general weights. The fastest algorithm for the case of Euclidean points in the plane, due to Vaidya [14], runs in $O(n^{2.5} \log^4 n)$ time. Special cases of the problem can be solved faster. For example, Marcotte and Suri [9] proposed an $O(n \log n)$ time exact algorithm for the case where the points ae the vertices of a convex polygon.

For large n, 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 shall mean the worst case ratio of the weight of an approximate

[1] This research was supported in part by the DIMACS Grant No. NSF-STC88-09648.
[2] This research was supported in part by the NSF under Grant No. CCR 88-07518.
Received November 1991. Revised November 1992.

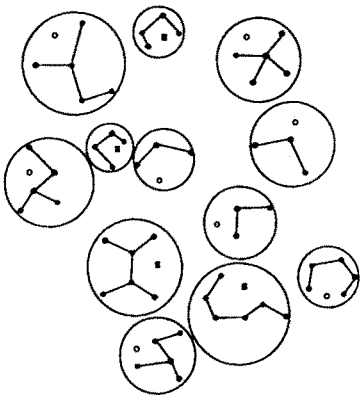

Fig. 1. The 1-basic graph with odd and even connected components (odd and even hypervertices).

solution produced by the heuristic to that of the optimal solution. For some perfect matching heuristics and lower bounds, see Grigoriadis and Kalantari [6, 7].

The *hypergreedy* heuristic for perfect matching due to Supowit, Plaisted, and Reingold [12], and Plaisted [11] runs in $O(n^2 \log n)$ time and obtains an approximate solution with error bound of $2\log_3 \frac{3}{2}n$. We propose a generalization of the hypergreedy, called the *t-hypergreedy*, where t is an integer parameter satisfying $1 \leq t \leq \lfloor \log_3 n \rfloor$, which provides an approximate solution with the error bounded above by $(2t + 1)$, and the time complexity of $O(\max\{tn^2, n^3 3^{-t}\})$. For $t = \lfloor \log_3 n \rfloor$ this heuristic reduces the hypergreedy.

We show that for points in the Euclidean plane the error of a modified t-hypergreedy is bounded above by $\alpha(2t + 1)$, where $\alpha = 2.42$, and its time complexity is $O(\max\{tn \log n, n^2 3^{-t} \log n, n^3 3^{-3t}\})$. In particular, for $t = \lfloor \log_3 n \rfloor$ the modified t-hypergreedy has an error bounded above by $\alpha(2\lfloor \log_3 n \rfloor + 1)$ and it can be implemented in $O(n \log^2 n)$ time. This time complexity is also favorable with respect to Vaidya's $O(n \log^3 n)$ time heuristic [15], for points in the Euclidean plane, whose error is bounded above by $3\log_3 \frac{3}{2}n$. The t-hypergreedy makes use of the *t-basic graph*, which is a collection of sparse connected components of $K(V)$. For points in the Euclidean plane the t-hypergreedy makes use of an approximate t-basic graph. The construction of these two graphs and the analysis of the error of the resulting t-hypergreedy heuristics are described in Section 2 and Section 3, respectively. In Section 4, we analyze the time complexity of the heuristics.

2. The t-basic graph and the t-hypergreedy heuristic.

In this section we describe the construction of the *t-basic graph*, a subgraph of $K(V)$, where t is an integer satisfying $1 \leq t \leq \lfloor \log_3 n \rfloor$. We deonote it by $BG_t(V)$. $BG_t(V)$ is obtained recursively from $BG_{t-1}(V)$ using edges in $K(V)$, and $BG_{t-1}(V)$ is

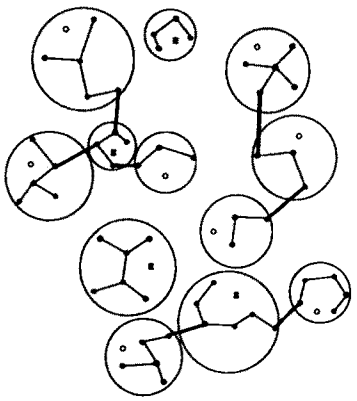

Fig. 2. The nearest neighbor graph of odd hypervertices.

a subgraph of $BG_i(V)$. To construct $BG_i(V)$ from $BG_{i-1}(V)$, we have to consider connected components with odd and even number of vertices. $BG_1(V)$ is the nearest neighbor graph of $K(V)$, i.e. each vertex in V will be connected to a nearest neighbor via the corresponding edge. In case of duplication of an edge only one copy is retained. Next we classify connected components in $BG_1(V)$ with respect to the number of vertices they span. Let $O_1(V)$ and $E_1(V)$ be the *odd* and *even* connected components of $BG_1(V)$, with odd and even number of vertices, respectively, see Fig. 1. We shall refer to the components as *hypervertices.*

$BG_2(V)$ is constructed by finding the nearest neighbor graph of the odd hypervertices in $BG_1(V)$, where each odd hypervertex in $O_1(V)$ is connected to a nearest odd hypervertex, either by an edge or a set of edges forming a shortest path in $K(V)$ passing through even hypervertices of $E_1(V)$, and again we keep only one copy of any duplicated edge, see Fig. 2. $BG_2(V)$ is a collection of old and new even hypervertices, $E_2(V)$, and new odd hypervertices, $O_2(V)$. In general, given $BG_i(V)$ we obtain $BG_{i+1}(V)$ by repeating the same process. The set of edges in $K(V)$ added in the process has total weight bounded above by the following lemma.

LEMMA 2.1. *For each $1 \leq i \leq t$, the total weight of all edges to $BG_i(V)$ to form $BG_{i+1}(V)$ is bounded above by $2\omega(M^*(V))$, where $M^*(V)$ and $\omega(M^*(V))$ denote an optimal perfect matching of V and its total edge weight, respectively. In particular, the weight of edges in $BG_t(V)$ is bounded above by $2t\omega(M^*(V))$.*

PROOF. For a given i, consider the union of the edges in the subgraph $BG_i(V)$ and those of $M^*(V)$. The edges of $M^*(V)$ partition the set of the odd hypervertices, $O_i(V)$, into pairs which are connected either by an edge in $M^*(V)$ or a chain of edges in $M^*(V)$ passing through even components of $E_i(V)$. In Fig. 3 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

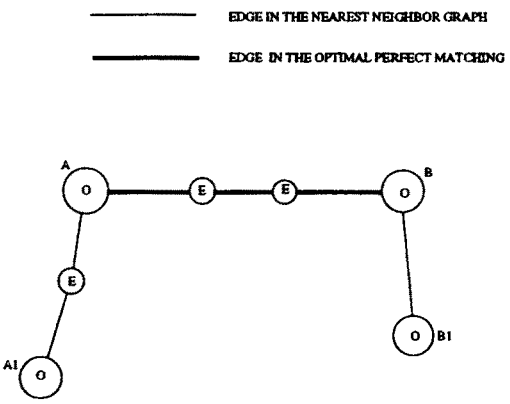

Fig. 3. A pair of odd hypervertices and their nearest odd neighbors.

equal to the weight of the chain of the edges in $M^*(V)$ 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 this chain.

Since the weight of each such chain formed by the edges in $M^*(V)$ is compared to the weight of at most two shortest paths connecting odd hypervertices, it follows that the total weight of the nearest neighbor graph of the odd hypervertices of $BG_t(V)$ is bounded above by 2 times the weight of $M^*(V)$. Hence the total weight of $BG_t(V)$ is bounded above by $2t$ times the weight of $M^*(V)$. ■

We now describe the t-hypergreedy heuristic. First, we construct the t-basic graph, $BG_t(V)$ This graph contains even and (possibly) odd hypervertices. If there are any odd hypervertices in the t-basic graph, we match them using an optimal perfect matching algorithm. This requires a preprocessing which consists of the computation of the shortest paths, possibly passing through even hypervertices, between every pair of odd hypervertices. Each shortest path is replaced by an edge which has the same weight as the weight of the path. This gives a complete graph whose nodes are the odd hypervertices. Next we compute an optimal perfect matching on the reduced graph. This type of problem reduction was exploited in Grigoriadis and Kalantari [7]. The matching is then replaced by the corresponding set of paths. The edges in the paths are added to $BG_t(V)$. The new graph contains only even connected components. Each component admits a perfect matching and is processed separately. The matching is obtained in a similar fashion as the minimum spanning tree heuristic for the traveling salesman problem [10]: The edges in each connected component are doubled. This results in a new graph, where each component is Eulerian. We extract an Euler tour in every connected component. An Euler tour is a closed walk where each edge is visited exactly once. Using the triangle inequality, we replace Euler tours with Hamiltonian cycles of lesser weight. A Hamiltonian cycle of a connected component is a cycle where each vertex is visited exactly once. Each Hamiltonian cycle is the union of two disjoint perfect matchings.

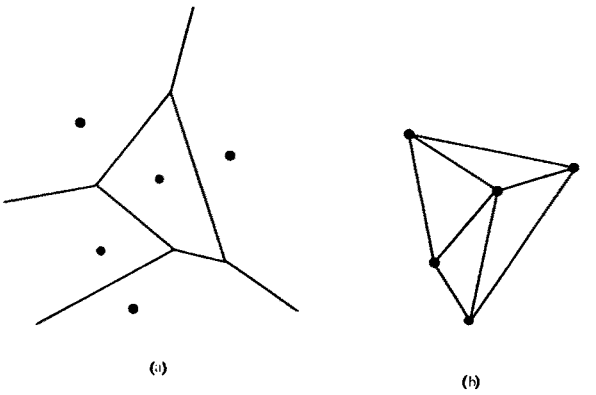

Fig. 4. (a) Voronoi diagram, (b) Delaunay triangulation.

We select the one with the smaller weight. The union of the selected perfect matchings of the components forms a perfect matching of V. The weight of the optimal perfect matching of the odd hypervertices is bounded above by $\omega(M^*(V))$.

In a similar fashion, as in the proof of Lemma 2.1, we consider the union of $M^*(V)$ and the optimal perfect matching of the odd hypervertices. This union gives even cycles on the odd hypervertices. Now it is easy to argue that in each cycle the total weight of edges from the optimal perfect matching of the odd hypervertices is less than or equal to the total weight of edges from $M^*(V)$. Thus

THEOREM 2.1. *The error of the t-hypergreedy for weights satisfying the triangle inequality, is bounded above by* $(2t + 1)$, *for* $1 \leq t \leq \lfloor \log_3 n \rfloor$. ■

3. An approximate t-basic graph and the t-hypergreedy for the Euclidean case.

Suppose V is a set of points in the Euclidean plane. In this case the t-hypergreedy builds *an approximate t-basic* graph, denoted by $\widehat{BG}_t(V)$, which is a subgraph of the *Delaunay triangulation*. For $i = 1$, the subgraph $\widehat{BG}_1(V)$ coincides with $BG_1(V)$, the nearest neighbor graph of $K(V)$. For $i > 1$, in obtaining $\widehat{BG}_{i+1}(V)$ from $\widehat{BG}_i(V)$, we construct the nearest neighbor graph of the odd hypervertices with respect to the Delaunay triangulation of V. This allows an efficient computation of an approximate nearest neighbor graph of the odd hypervertices within a constant error bound.

The Delaunay triangulation, which is a sparse graph with $O(n)$ edges, is obtained from the *Voronoi diagram*, see Fig. 4(a). The Voronoi diagram is a partition of the plane into *Voronoi regions* of points closer to one point of V than to others, see Preparata and Shamos [13]. The Voronoi region of a point $v \in V$ is a set of points in the plane closer to v than to any other point $u \in V$, $v \neq u$. Two Voronoi regions are

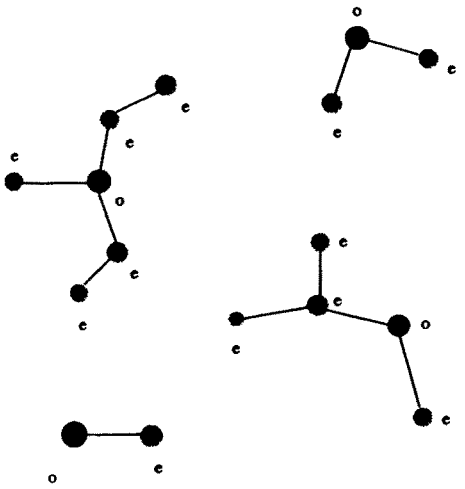

Fig. 5. Generalized Voronoi diagram.

adjacent if they share a Voronoi edge. The Delaunay triangulation of V can be derived from the corresponding Voronoi diagram by joining each two points whose Voronoi regions are adjacent, see Fig. 4(b), in $O(n)$ time. Chew [1] and Dobkin and Friedman [3] showed that complete Euclidean graphs can be approximated by Delaunay graphs. The best bound on this approximation has been recently shown by Keil and Gutwin in [8], proving that the Delaunay triangulation has the property that the weight of the shortest path between each pair of vertices is bounded above by $\alpha = 2.42$ times the Euclidean distance between the two vertices.

Consider an odd hypervertex A. Let B be the nearest odd hypervertex of A with respect to $K(V)$, and let $(v_1, v_2), (v_3, v_4) \ldots, (v_{k-1}, v_k)$ be the edges of this shortest path. From the above, for each edge (v_i, v_{i+1}), there exists a path in the Delaunay triangulation whose weight is bounded above by α times the straight line distance between v_i and v_{i+1}. Thus

LEMMA 3.1. *The weight of the shortest path from an odd hypervertex to the nearest odd hypervertex with respect to the Delaunay triangulation, is bounded above by α times the weight of the shortest path from that odd hypervertex to its nearest odd hypervertex with respect to $K(V)$.*

Using Lemma 2.1 and the above lemma, we conclude the following for points in the Euclidean plane.

COROLLARY 3.1. *The weight of edges of $\widehat{BG}_t(V)$ is bounded above by $2\alpha t\omega(M^*(V))$.*

The approximation of $BG_t(V)$ by $\widehat{BG}_t(V)$ results in a corresponding approximate t-hypergreedy, for points in the Euclidean plane, which satisfies the following

THEOREM 3.1. *The error of the approximate t-hypergreedy for points in the Euclidean plane is bounded above by* $\alpha(2t+1)$, *for* $1 \leq t \leq \lfloor \log_3 n \rfloor$.

4. Time complexities.

In this section we analyze the worst case time complexities for constructing $BG_t(V)$, its approximation $\widehat{BG}_t(V)$ for points in the Euclidean plane, and the corresponding heuristics.

As in the case of *hypergreedy* [11] the construction of the nearest neighbor graph of odd hypervertices consists of two stages. In the first stage, we construct 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, see Fig. 5. Then, in the second stage, a nearest odd neighbor is found for each odd hypervertex. To construct the *GVD*, for every hypervertex we find a shortest path to its nearest odd hypervertex, possibly passing through other even hypervertices. Each odd hypervertex and all the even hypervertices, which are closer to it than to any odd hypervertex, form a *Generalized Voronoi Region*, (*GVR*). For general weights satisfying the triangle inequality, *GVD* is constructed using edges in $K(V)$, while for points in the Euclidean plane *GVD* only edges in the Delaunay triangulation are used. For the Euclidean case two *GVR*'s are *adjacent* if there is an edge in the Delauney triangulation with endpoints in the *GVR*'s. For complete graphs each two *GVR*'s are adjacent because there is always an edge in $K(V)$ with endpoints in the *GVR*'s.

The construction of $BG_1(V)$ clearly requires $O(n^2)$ operations in the general case. For the Euclidean case, the construction of the Voronoi diagram of V requires $O(n \log n)$ time, where $\widehat{BG}_1(V)$ which coincides with $BG_1(V)$ can be obtained in $O(n)$ operations, see [13].

Now consider the following problem. Let $G = (V, E)$ be a weighted graph with $|V| = n$ vertices and $|E| = m$ edges and whose weights are nonnegative. Suppose that V is partitioned into A and B labeled as *odd* and *even* vertices, respectively. If we define *GVD* as before, i.e. as clusters of even vertices connected to their nearest odd vertices by the corresponding shortest paths, then we have

LEMMA 4.1. *The GVD of* $G = (V, E)$ *can be constructed in* $O(\min\{n^2, m \log n\})$ *time.*

PROOF. Consider the following modified Dijkstra's shortest path algorithm (see [10], [11]).

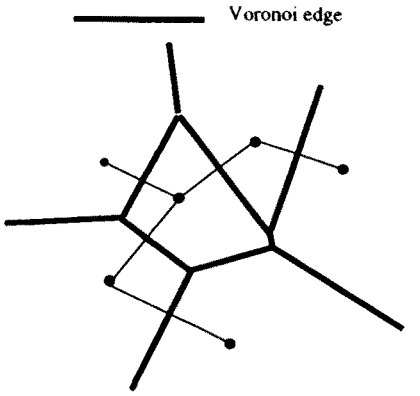

Fig. 6. Voronoi region of a hypervertex (VRH).

Dijkstra's modified algorithm for GVD.

Input: A graph $G = (V, E)$ with nonnegative edge weights and with V partitioned into two subsets A and B, where $A = \{\text{vertices labeled as even}\}$ and $B = \{\text{vertices labeled as odd}\}$. Initially each vertex $v \in A$ has a label $p(v) = \infty$.

Output: The weight of the shortest path from each even vertex v to its nearest odd vertex, possibly passing through intermediate even vertices, stored in $p(v)$. Also each even vertex will store the next vertex in its shortest path.

begin
  for all $v \in A$ such that there is an edge $(v, o) \in E$, where $o \in B$ **do**
    $p(v) = min\{\omega(v, o)\colon o \in B \text{ and } (v, o) \in E\}$;
  while $A \neq \emptyset$;
  begin find $min\{p(v); v \in A\}$, say $p(q)$;
    $B = B \cup \{q\}$;
    $A = A \backslash \{q\}$;
    for all $v \in A$ s.t. $(v, q) \in E$ **do**
      $p(v) = min\{p(v), p(q) + \omega(v, q)\}$
  end
end

For $G = (V, E)$ dense, there are n iterations in the main loop of the algorithm, where each iteration is proportional to the number of vertices in A. Thus, the time complexity is $O(n^2)$. For sparse graphs, we can store all the labels $p(v)$ on a heap to maintain a priority queue which returns and removes the label with the smallest value, see [2]. The size of the heap is $O(n)$ and it can be constructed in $O(n)$ time. The algorithm will require m operations on the priority queue where each operation takes $O(\log n)$ time. Thus the time complexity is $O(m \log n)$. ■

The following was proved in [11].

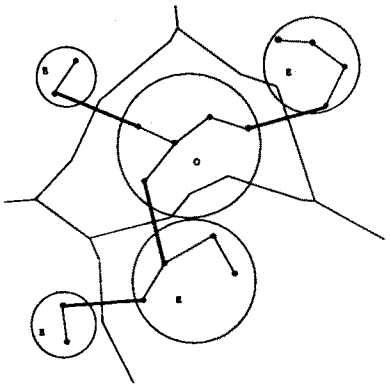

Fig. 7. Representation of a *GVR* as a cluster of *VRH*'s.

LEMMA 4.2. *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 the 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 *GVD* we can determine in $O(n^2)$ operations the nearest odd neighbor for each odd hypervertex in $BG_1(V)$ with respect to $K(V)$. We examine all the edges in $K(V)$ and regard those with endpoints in different *GVR*'s, and select for each pair of *GVR*'s such an edge which minimizes the length of the shortest path between the corresponding odd hypervertices.

For points in the Euclidean plane, if instead of $K(V)$ we consider the nearest odd neighbor with respect to the Delaunay triangulation, then the above Lemma implies that the computation of the nearest odd neighbors, given *GVD*, can be done in $O(n)$ operations. Each vertex from the set V defines its Voronoi region of points closer to it than to any other vertex in V. We represent each hypervertex, odd and even, of $BG_1(V)$, by a cluster of Voronoi regions of all the vertices in the hypervertex. We denote this representation by the *Voronoi region of a hypervertex*, (*VRH*), see Fig. 6.

Two *VRH*'s are *adjacent* if at least one pair of their constituent Voronoi regions, each in different *VHR*, is adjacent. We represent each *GVR* as a cluster of *VRH*'s of all the hypervertices in the *GVR*, see Fig. 7. Two *GVR*'s are *adjacent* if at least one pair of their constituent *VRH*'s, each in different *GVR*, is adjacent. To find for each odd hypervertex its nearest odd hypervertex with respect to the Delaunay triangulation, given the above representation of *GVD*, see Fig. 8, we examine all the edges in the Delaunay triangulation and consider those edges with endpoints in different *GVR*'s. Then we select for each pair of adjacent *GVR*'s such an edge which minimizes the length of the shortest path between the corresponding odd hypervertices. Thus

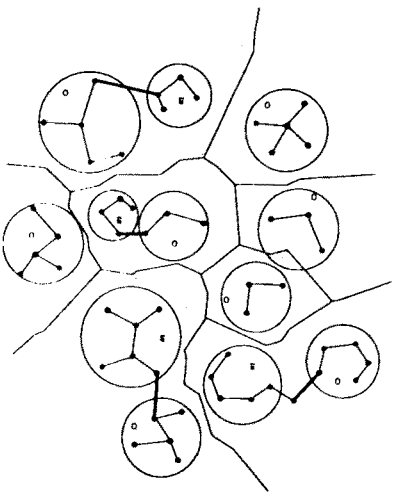

Fig. 8. Generalized Voronoi diagram with its GVR's represented as clusters of VRH's.

THEOREM 4.1. *The nearest neighbor graph of odd hypervertices can be computed in* $O(n^2)$ *time with respect to* $K(V)$, *and in* $O(n \log n)$ *time for points in the Euclidean plane with respect to the Delaunay triangulation.*

To construct $BG_i(V)$ from $BG_{i-1}(V)$, we have to find for every hypervertex its nearest odd hypervertex. By Theorem 4.1, this requires $O(n^2)$ and $O(n \log n)$ time for general complete graphs and for points in the Euclidean plane, respectively. Thus

COROLLARY 4.1. *The time complexity of constructing* $BG_t(V)$ *is* $O(tn^2)$. *For points in the Euclidean plane the time complexity for constructing* $\widehat{BG}_t(V)$ *is* $O(t|V| \log |V|)$.

We now analyze the time complexity of the second stage of the t-hypergreedy, i.e. the time needed to find the optimal perfect matching of the odd hypervertices either in $BG_t(V)$ or $\widehat{BG}_t(V)$. There is an even number of odd hypervertices in $BG_t(V)$ or $\widehat{BG}_t(V)$, if any. Having found an optimal perfect matching of the odd hypervertices, we add the corresponding edges. This will result in a graph with hypervertices which are even only.

For each t, the number of vertices in each odd hypervertex of $BG_t(V)$ or $\widehat{BG}_t(V)$ is at least 3^t. This follows from the fact that at each stage a new odd hypervertex is formed by an odd number of odd hypervertices (at least 3 of them); thus the number of vertices in an odd hypervertex grows by a factor of at least 3. In particular, when $t = \lfloor \log_3 n \rfloor$ the heuristic reduces to the hypergreedy heuristic. For this t, there could be at most two remaining odd hypervertices. Otherwise, the graph would have at

least $3^{\lfloor \log_3 n \rfloor + 1}$ vertices, which is impossible. Thus, for this t, the construction of the second part of the t-hypergreedy, i.e., the construction of the optimal perfect matching of the odd hypervertices is essentially the construction of the shortest path between the two odd hypervertices. This can be done in $O(n^2)$ time for general weights and $O(n \log n)$ time for the Euclidean case.

Given $BG_t(V)$, where odd and even hypervertices are treated as nodes, the t-hypergreedy proceeds as follows. We form a graph with those edges in $K(V)$, whose endpoints are in different hypervertices. In this graph, for all its $O(n3^{-t})$ odd nodes, the shortest paths between every pair of vertices is found. This can be done in $O(n^3 3^{-t})$ time by applying the shortest path algorithm of Dijkstra $O(n3^{-t})$ times. We replace each such shortest path by an edge whose weight is equal to that of the path, and we match odd hypervertices in the complete graph using an exact perfect matching algorithm, in $O(n^3 3^{-3t})$ time. Now all the connected components are even. The construction of Euler and Hamiltonian cycles requires $O(n)$ time. Each Hamiltonian gives rise to two different perfect matchings on its vertices, and the better of the two is selected. Thus we have

THEOREM 4.2. *The time complexity of the t-hypergreedy is* $T(n) = O(\max\{tn^2, n^3 3^{-t}\})$, *where* $1 \leq t \leq \lfloor \log_3 n \rfloor$.

Given $\widehat{BG}_t(V)$ for points in the Euclidean plane, the second part of the modified t-heuristic is implemented as follows. We form a graph where nodes are all even and odd hypervertices of $\widehat{BG}_t(V)$, and they are connected by those edges in the Delaunay triangulation whose endpoints are in different hypervertices. In this graph for each odd hypervertex we find a shortest path to all other odd hypervertices in $O(n^2 3^{-t} \log n)$ time using Dijkstra's shortest path algorithm $O(n3^{-t})$ times. As before, we form a complete graph whose nodes are all the odd hypervertices and edges correspond to the shortest paths between the hypervertices, followed by the computation of an optimal perfect matching. Thus we have

THEOREM 4.3. *For points in the Euclidean plane, the time complexity of the modified t-hypergreedy is* $T(n) = O(\max\{tn \log n, n^2 3^{-t} \log n, n^3 3^{-3t}\})$, *where* $1 \leq t \leq \lfloor \log_3 n \rfloor$.

Acknowledgement.

We wish to thank the referees for their constructive comments. In particular, we thank one of the referees for making us aware of the recent result by Keil and Gutwin [8] which allowed us to improve the error bound of our heuristic, originally based on Chew's result [1].

REFERENCES

1. L. P. Chew, *There is a planar graph almost as good as the complete graph*, Proceedings of the 2nd Symposium on Computational Geometry, Yorktown Heights, NY, 169–177, (1986).
2. T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, The MIT Press and McGraw-Hill, Cambridge, (1989).
3. D. P. Dobkin, S. J. Friedman and K. J. Supowit, *Delaunay graphs are almost as good as complete graphs*, FOCS, 20–26, (1987).
4. J. Edmonds, *Paths, trees and flowers*, Canadian Journal of Mathematics, (17), 449–467, (1965).
5. H. N. Gabow, *Implementation of algorithms on nonbipartite graphs*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, (1973).
6. M. D. Grigoriadis and B. Kalantari, *A lower bound to the complexity of Euclidean and rectilinear matching algorithms*, Information Processing Letters, (22), 73–76, (1986).
7. M. D. Grigoriadis and B. Kalantari, *A new class of heuristic algorithms for weighted perfect matching*, JACM, (35), 769–776, (1988).
8. J. M. Keil and C. A. Gutwin, *Classes of graphs which approximate the complete Euclidean graphs*, Discrete and Computational Geometry, (7), 13–28, (1992).
9. O. Marcotte and S. Suri, *Fast matching algorithms for points on a polygon*, FOCS, 60–65, (1989).
10. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, New Jersey, (1982).
11. D. A. Plaisted, *Heuristic matching for graphs satisfying the triangle inequality*, Journal of Algorithms, 5 (5), 163–179, (1984).
12. K. J. Supowit, D. A. Plaisted and E. M. Reingold, *Heuristic for weighted matching*, STOC, 398–419, (1980).
13. F. P. Preparata and K. J. Shamos, *Computational Geometry*, Springer-Verlag, New York, (1985).
14. P. Vaidya, *Geometry helps in matching*, (extended abstract), STOC, 422–425, (1988).
15. P. Vaidya, *Approximate minimum weight matching on points in k-dimensional space*, Algoritmica, 4, 569–583, (1989).