Finding a Cheap Matching

T H C Smith*
Department of Computer Science, Rand Afrikaans University, P O Box 524, Johannesburg, 2000

Abstract

We propose an edge exchange algorithm for finding a cheaper 1-matching from a given 1-matching as well as a heuristic algorithm for constructing a relatively cheap 1-matching from the optimal solution of a relaxation of the 1-matching problem. Computational experience with these two algorithms as well as two greedy algorithms from the literature is reported.
Keywords: *matchings, heuristics, approximate algorithms, empirical analysis*

Received January 1988, Accepted September 1988

1. Introduction

Let G be a complete, undirected graph with node set N = {1, 2,..., n} and edge set E = {1, 2,..., m} where n is even and m = n(n – 1)/2. A *perfect matching* (or *1-matching*) in the graph G is a subset M of n/2 edges such that exactly one edge in M meets any node. If each edge e in E has an associated integer cost c_e, then the *perfect matching problem*, PMP, is to find a perfect matching M for which the total cost of the edges in M is minimized.

Exact algorithms for solving PMP have been known for several decades (a FORTRAN implementation of such an algorithm is given by Burkhard & Derigs [1]). However, these algorithms have a run time of $O(n^3)$ which makes them too slow for large-scale problems. Therefore several heuristic algorithms have been proposed (for example by Reinhold & Tarjan [6] and Grigoriadis et al [4]) for finding a relatively cheap 1-matching faster than finding the optimal matching with an exact algorithm. Although the worst-case behaviour of the proposed heuristic algorithms has been analysed, very little computational experience with these algorithms has been reported.

In this paper we propose an algorithm for finding a cheaper 1-matching from a given 1-matching as well as a new heuristic algorithm for finding a cheap 1-matching. We then report our computational experience with these two algorithms as well as two other heuristic algorithms.

2. An Edge Exchange Algorithm

The use of edge exchange algorithms in heuristic algorithms for the travelling salesman problem is well known (see Golden et al [3]). In this section we propose an edge exchange algorithm for finding a cheaper 1-matching from a given 1-matching, if possible.

If (a,b) and (c,d) are two edges in a 1-matching, then a new 1-matching can be obtained by exchanging these two edges either with the two edges (a,c) and (b,d) or with the two edges (a,d) and (b,c). We call this type of edge exchange an *improving 2-exchange* if it results in a cheaper 1-matching. A 1-matching is called *2-optimal* if it does not admit an improving 2-exchange.

Given a 1-matching, the following multipass algorithm, which we shall call IMPROVE, can be used to obtain a 2-optimal 1-matching: In the first pass all n(n-2)/8 edge pairs in the current 1-matching are tested for a potential improving 2-exchange (when an edge pair allowing an improving 2-exchange is found, the 2-exchange is carried out immediately). In any subsequent pass only edge pairs involving at least one edge replaced in the previous pass need to be evaluated for an improving 2-exchange. The algorithm terminates when no improving 2-exchange is found in a new pass.

3. A Linear Programming Heuristic

PMP can be formulated as the following linear programming problem:

$$\begin{array}{ll} \text{minimize} & \underline{c}\,\underline{x} \\ \text{subject to} & A\underline{x} = \underline{1} \\ & \underline{x} \in X \\ & \underline{x} \geq \underline{0} \end{array}$$

where A denotes the n x m incidence matrix of the

* The research was done as visiting scientist at the National Research Institute for Mathematical Sciences, Council for Scientific and Industrial Research, South Africa.

graph, $\underline{0}$ is the zero m-vector, $\underline{1}$ is an n-vector with all elements equal to 1, the m-vectors $\underline{c}$ and $\underline{x}$ respectively contain the costs and decision variables for the edges, and X is a polyhedral set (i.e. a set defined by a number of linear inequalities).

The problem RPMP obtained from PMP by relaxing (ignoring) the constraint $\underline{x} \in X$ can be formulated as an assignment problem on a bipartite graph with 2n nodes (Derigs and Metz [2] used this approach to get a good initial solution in an exact matching algorithm). However, RPMP is a generalized network problem that can be solved efficiently using a specialized primal simplex algorithm (see Kennington & Helgason [5]). In a basic feasible solution of RPMP $x_e \in \{0, 1/2, 1\}$ for all $e \in E$, and the set of edges $\{e \in E : x_e = 1/2\}$ forms an even number of cycles each containing an odd number of nodes.

We shall now describe a heuristic algorithm, called LPMATCH, which constructs a 1-matching from an optimal basic feasible solution of RPMP. The first part of LPMATCH constructs a partial 1-matching:

```
solve RPMP using the primal simplex algorithm
M := {e ∈ E : x_e = 1}; P := {e ∈ E : x_e = 1/2}
for each cycle in P do
        i := node with largest dual value in cycle
        S := {cycle edges matching all nodes in
                cycle except i}
        M := M ∪ S
end
```

The total cost of any set of cycle edges matching all nodes in a cycle except one equals the sum of the dual values of the matched nodes in the cycle. Hence, the rule in the above loop for selecting the unmatched node i in a cycle ensures that the total cost of the edges in the set S is minimized.

The second part of LPMATCH completes the partial 1-matching M. Since there is not much freedom left in selecting the additional edges, a preliminary improvement phase is incorporated to lessen the effect of bad selections in this part of LPMATCH:

```
M' := M
while |M| < n/2 do
        select an unmatched node i
        e := cheapest edge between i and another
                unmatched node
        M := M + {e}
end
for each e ∈ M – M' do
        if there is an improving 2-exchange
                involving e then perform best
                improving 2-exchange involving e
        end
end
```

4. Computational Comparison

The greedy heuristic algorithm, which we call GREEDY, has been analyzed in [6]. It constructs a 1-matching by repeatedly selecting the cheapest edge connecting two unmatched nodes. Weaker versions of the greedy algorithm have been investigated in [4]. One of these versions, which we call S-GREEDY, constructs a 1-matching by repeatedly selecting an unmatched node i and then selecting the cheapest edge between i and another unmatched node.

We have implemented the algorithms IMPROVE, LPMATCH, S-GREEDY and GREEDY in one Pascal program. In the program each of the algorithms S-GREEDY, GREEDY and LPMATCH is used to obtain a 1-matching, each of which is then improved using the algorithm IMPROVE.

In the implementation of algorithm LPMATCH the 1-matching found by algorithm S-GREEDY is used to construct an initial basic feasible solution for RPMP. A generalized network code is then used to obtain an optimal basic feasible solution for RPMP.

The program was compiled with the Pascal-6000 compiler (Version 4.1) and executed on a CDC 750 computer running the NOS 4.2 operating system. The run times reported below were obtained with a program in which all run time checks were turned off.

We tested the algorithms using 62 Euclidean problems and 30 non-Euclidean problems. The Euclidean problems ranged in size from 60 to 200 nodes and were generated by randomly selecting points in the twodimensional plane with both coordinates between 1 and 1000. The non-Euclidean problems ranged in size from 60 to 100 nodes and were generated by selecting the edge costs randomly between 1 and 1000. For each problem size not larger than 100 nodes ten problems were generated, while three problems were generated for each larger problem size.

The average ratios between the costs of the 1-matchings found by the different algorithms and the cost of an optimal 1-matching (obtained using the FORTRAN code, SMP, in [1]) are reported in Table 1. The table also contains the average ratios for the complete set of Euclidean problems. In the table the problem type is identified as either E (for Euclidean) or N (for non-Euclidean).

For non-Euclidean problems LPMATCH gives much better results than S-GREEDY and GREEDY (which both perform poorly). The 1-matchings constructed by S-GREEDY and GREEDY are greatly improved by IMPROVE while the near-optimal 1-matchings constructed by LPMATCH are only slightly improved.

In the case of the Euclidean problems LPMATCH also constructs much better 1-matchings than S-GREEDY and GREEDY. The algorithm IMPROVE produces near-optimal 1-matchings from the 1-

matchings constructed by S-GREEDY, LPMATCH and GREEDY.

The average run times for the different heuristic algorithms are reported in Table 2. We used the least squares method to fit a polynomial function of the form $t(n) = an^b$ to the average run times of each of the three combinations of S-GREEDY, LPMATCH and GREEDY with IMPROVE as well as to the average run times of the exact code SMP. The estimates of a and b, as well as the coefficient of determination r^2 for each fit, are reported in Table 3. The observed run time of each of the combinations has an approximately quadratic growth rate (although the observed run time of LPMATCH+IMPROVE grows slower than that of GREEDY+IMPROVE).

Since the run time of a Pascal program on the CDC 750 computer is at least double the run time of an equivalent FORTRAN program, the run time of the exact code SMP cannot be compared directly with that of the heuristic algorithms. However, it should be noted that the observed run time of SMP grows much faster, at almost a cubic rate (see Table 3).

5. Conclusion

We have proposed a heuristic algorithm based on a relaxation of the perfect matching problem that performs better than the semi-greedy and greedy heuristic algorithms on both Euclidean and non-Euclidean problems. The edge exchange algorithm that we devised for improving on a given 1-matching produces near-optimal 1-matchings without great computational effort. The observed run times of these heuristic algorithms grow much slower than the run time for the exact algorithm.

References

[1] R E Burkard and U Derigs, [1980], Assignment and Matching Problems: Solution Methods with FORTRAN Programs, *Lecture Notes in Economics and Mathematical Systems*, **184**, Springer, Berlin.

[2] U Derigs and A Metz, [1986], On the use of optimal fractional matchings for solving the (integer) matching problem, *Computing*, **36**, 263-270.

[3] B Golden, L Bodin, T Doyle and W Stewart, [1980], Approximate traveling salesman algorithms, *Operations Research*, **28**, 694-711.

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

[5] J L Kennington and R V Helgason, [1980], *Algorithms for Network Programming*, John Wiley and Sons, New York.

[6] E M Reinhold and R E Tarjan, [1981], On a greedy heuristic for complete matching, *SIAM Journal on Computing*, **10**, 676-681.

TABLE 1
Average ratios

Type	Nodes	S-GREEDY	IMPROVE	LPMATCH	IMPROVE	GREEDY	IMPROVE
E	60	1,325	1,033	1,097	1,006	1,226	1,016
E	70	1,422	1,047	1,078	1,021	1,275	1,035
E	80	1,322	1,065	1,120	1,023	1,218	1,040
E	90	1,378	1,033	1,108	1,028	1,200	1,024
E	100	1,403	1,060	1,096	1,020	1,204	1,017
E	130	1,479	1,086	1,066	1,018	1,277	1,015
E	150	1,461	1,093	1,104	1,007	1,285	1,012
E	180	1,489	1,094	1,134	1,021	1,276	1,045
E	200	1,369	1,066	1,121	1,034	1,261	1,049
Average of E		1,385	1,055	1,099	1,020	1,234	1,027
N	60	2,637	1,639	1,039	1,033	2,234	1,517
N	70	2,619	1,593	1,033	1,028	2,366	1,514
N	100	2,592	1,671	1,026	1,021	2,556	1,596

TABLE 2
Average run times (in milliseconds)

Type	Nodes	S-GREEDY	IMPROVE	LPMATCH	IMPROVE	GREEDY	IMPROVE
E	60	5,7	52,5	155,7	43,5	163,7	46,0
E	70	7,7	76,8	209,8	56,7	229,7	67,6
E	80	9,8	94,9	285,1	76,0	300,9	83,3
E	90	12,6	128,6	330,7	97,9	381,9	103,3
E	100	14,9	169,8	416,1	116,1	477,2	127,2
E	130	23,3	260,3	617,3	186,3	788,7	226,3
E	150	33,0	339,7	896,7	278,3	1096,0	304,0
E	180	45,3	496,3	1211,3	375,7	1625,0	439,7
E	200	55,7	604,3	1497,0	457,7	1990,3	521,7
N	60	6,1	45,8	207,4	29,8	162,1	44,0
N	70	8,1	63,3	295,5	39,6	225,5	64,8
N	100	14,8	126,7	602,8	83,2	457,6	132,4

TABLE 3
Estimates for polynomial fit

	S-GREEDY+IMPROVE	LPMATCH+IMPROVE	GREEDY+IMPROVE	SMP
a	0,02365	0,09037	0,04662	0,00398
b	1,921	1,883	2,057	2,676
r^2	0,9809	0,9983	0,9998	0,9966