Mathematical Programming 50 (1991) 113-121
North-Holland

Solving (large scale) matching problems combinatorially

U. Derigs and A. Metz

Lehrstuhl für Wirtschaftsinformatik, Universität zu Köln, W-5000 Köln, Germany

Received 14 May 1987
Revised manuscript received March 1989

In this paper we describe computational results for a modification of the shortest augmenting path approach for solving large scale matching problems. Using a new assignment start procedure and the two-phase strategy, where first the problem is solved on a sparse subgraph and then reoptimization is used, matching problems on complete graphs with 1000 nodes are solved in about 10–15 seconds on an IBM 4361.

Key words: Matching problem, two-phase approach, fast matching algorithms.

1. Introduction

The problen of finding a min-cost perfect matching in a graph $G=(V, E)$ is one of the few efficiently solvable cornerstone problems in combinatorial optimization. Until recently all efficient matching algorithms were combinatorially natured and based on the principle of iteratively constructing shortest (augmenting) paths, i.e. following the classical ideas of Edmonds (1965).

Recently Grötschel and Holland (1985) presented results with an LP-based facet generating cutting plane algorithm. For large problems their implementation which uses a commercial LP-package — the system MPSX from IBM — was at least comparable to the shortest augmenting path implementation SMP published in Burkard and Derigs (1981).

In this note we briefly analyze Grötschel and Holland's implementation and we describe its main ideas and their limitations. We then outline a new SMP-based implementation which outperforms its competitors by an order of magnitude.

In the following we assume that the reader is familiar with the problem of matchings in graphs and thus we avoid the usual catalog of lengthy definitions.

2. The facet-generating approach

The basic idea of this approach is the following. Starting from the LP-relaxation of

This work was partially supported by Sonderforschungsbereich 303, University of Bonn, and a special grant from the Deutsche Forschungsgemeinschaft.

the matching problem

$$\min\{c'x \mid Ax = 1, x \geqslant 0\}$$

with A the node-edge incidence matrix of G, a sequence of linear programs is solved by the LP-solver MPSX and its reoptimization features. This sequence is defined by successively adding facets of the matching polytope, i.e. so-called blossom constraints of the form

$$\sum_{i,j \in W} x_{i,j} \leqslant \tfrac{1}{2}(|W| - 1) \quad \text{where } W \subset V \text{ with } |W| \geqslant 3, \text{ odd},$$

to the LP-relaxation which "cut-off" the optimal fractional solution of the predecessor problem until an optimal integer solution is obtained which then defines an optimal matching.

Grötschel and Holland's (1985) results show that the number of facets to be added is extremely small. Also, the blossom constraints which cut-off the noninteger optima are constructed via efficient heuristics, a fact that speeds up the whole solution process.

The key-issue in Grötschel and Holland's implementation of the facet-generating approach to the matching problem is the following. Instead of considering the whole graph throughout the solution process they start the procedure on a *sparse* subgraph G' which is very likely to contain an optimal matching of G. This subgraph is constructed by only looking at the cheapest edges in the neighbourhood of any node. After the optimal matching in G' is constructed the remaining edges are checked for dual-feasibility ("outpricing-step"). Then either the present matching turns out to be optimal in G, too, or the edges which do not price out correctly — i.e. those which are dual infeasible — are added to G' and a new optimal matching in the modified subgraph G' is constructed.

Keeping the working subgraph small this way speeds up the time for the LP-solver significantly. Moreover the results of Grötschel and Holland indicate that on their randomly generated problems choosing the cheapest 5 edges in the neighbourhood of any node, the optimal matching in the initial subgraph G' often occurs to be optimal in G, too.

This approach of first solving a smaller subproblem and then performing reoptimization introduced by Holland and Grötschel (1985) for the matching problem is a common technique in linear programming and has been applied successfully to the assignment problem by Gavish et al. (1977).

3. The new SMP-based approach

During the last years we have improved the basic SMP-approach in two respects, too, by

- developing a new startprocedure, and
- implementing the idea of keeping the working subgraph sparse.

With these modifications the combinatorial approach has become significantly faster.

Derigs (1985) has shown that reoptimizing a matching problem after some cost changes can be done by applying the basic shortest augmenting path idea and hence the existing efficient SMP-code. Thus a so-called "two-phase approach" which starts by solving a matching problem on a sparse subgraph and then eventually reoptimizes can be implemented using the basic SMP-instruments only. Several reoptimization strategies are described in Derigs (1986).

In Derigs and Metz (1986a) we have shown how the optimal primal and dual solution of the standard LP-relaxation of the matching problem can be used to construct a start solution for the shortest augmenting path method. Computational results have shown that the SMP-code, started with such an initial solution, outperforms the basic SMP-code significantly.

The LP-relaxation is thereby interpreted as an assignment problem on a related bipartite graph and solved by special assignment codes. Note that the assignment problem is a matching problem on a bipartite graph and can be solved by the two-phase approach based on the shortest augmenting path variant for bipartite matching problems.

Now we are able to outline the new SMP-based matching approach:

Two-phase approach for solving large scale matching problems.

PREPHASE: Solve the LP-relaxation using the two-phase assignment code.
From the optimal LP-solution construct a startsolution M_b for SMP.

PHASE I: Construct a sparse subgraph $G' = (V, E')$.
Solve the matching problem in $G' = (V, E \cup M_b)$ using a sparse version of the SMP-code starting from the initial matching M_b.

PHASE II: Check the optimal solution from PHASE I with respect to optimality in G ("outpricing step") and reoptimize using the SMP-code if necessary.

4. Computational results

Our computational results reported here were obtained on an IBM 4361 of the Sonderforschungsbereich 303 at the Institute for Operations Research at the University of Bonn. All running times — except for Table 3 — are given in CPU-seconds.

Our first set of data/problems contains complete graphs with $n = 600, 700, \ldots,$ 1000 nodes the (integer) cost coefficients of which were generated randomly (equally distributed) in the interval $[0, C]$ with $C = 100$, 1000, 10 000, 100 000. For each (n, C)-combination 10 examples are generated and contained in this set.

In Table 1 we display results for the basic (one-phase) shortest augmenting path approach. Here we elaborate on the original SMP-code contained in Burkard and Derigs (1980) where we only replaced the cost matrix by a simple list structure to

Table 1

Running times for SMP with different startprocedures

C	n	Subroutine SMP (MPS1K)		Subroutine SMP (MPASS		Subroutine SMP (MP2PA)	
		Average	Maximum	Average	Maximum	Average	Maximum
100	600	239.831	475.109	93.714	429.375	4.408	5.574
	700	294.393	552.504	14.529	16.713	24.875	77.526
	800	397.147	629.701	55.355	195.720	47.894	191.267
	900	910.769	1650.645	17.793	19.386	21.724	57.950
	1000	889.002	1552.638	26.780	35.317	17.925	19.755
1000	600	231.335	343.802	21.848	30.741	13.332	34.897
	700	358.701	717.770	39.042	83.203	17.595	55.774
	800	394.210	656.075	45.303	53.747	16.894	27.556
	900	755.683	1461.371	81.088	140.475	42.993	111.804
	1000	1523.564	2265.819	64.831	96.415	27.733	50.765
10 000	600	224.360	273.215	24.416	27.526	7.499	8.636
	700	398.987	454.618	60.030	135.812	26.442	98.525
	800	447.215	500.245	54.356	69.658	16.625	34.059
	900	938.202	1143.301	70.398	84.099	19.226	34.591
	1000	1042.907	1421.668	94.706	102.855	23.456	40.745
100 000	600	260.548	355.004	27.939	32.105	9.905	14.294
	700	371.148	416.912	39.803	43.507	10.116	13.169
	800	490.848	552.987	71.399	114.470	33.022	66.101
	900	1010.319	1587.376	69.966	81.942	18.714	31.985
	1000	1043.135	1303.158	97.801	127.786	24.908	31.420

represent the graph and the cost data. We tested this approach/code with three different startprocedures:

MPS1K: The original greedy-like startprocedure of the SMP-code from Burkard and Derigs (1980).

MPASS: The assignment startprocedure with solving the assignment problem via the (one-phase) shortest augmenting path method.

MP2PA: The assignment startprocedure with solving the assignment problem via the two-phase approach.

The message of this experiment is clear. The assignment start is significantly superior, especially when using the two-phase solution technique.

In Table 2 we present our results for the new two-phase matching code TPHASE on the same set of problems. The initial subgraph G' was constructed by choosing the $k=6$ resp. $k=8$ cheapest edges in the neighbourhood of any node — values which have shown to be rather efficient.

Table 2

Running times for the new two-phase matching approach

C	n	Subroutine TPHASE $k=6$		Subroutine TPHASE $k=8$	
		Average	Maximum	Average	Maximum
100	600	3.540	3.684	3.367	3.488
	700	4.376	5.158	4.329	4.709
	800	5.238	5.461	5.154	5.252
	900	7.113	9.159	6.918	8.377
	1000	8.978	11.741	8.858	10.919
1000	600	4.345	4.766	4.333	4.892
	700	5.849	6.982	5.907	7.518
	800	7.930	10.703	7.320	8.217
	900	9.911	12.646	9.640	11.249
	1000	10.621	12.137	10.906	12.986
10 000	600	4.554	4.692	4.594	4.799
	700	6.684	9.328	6.558	9.275
	800	7.769	8.307	7.725	7.937
	900	9.816	12.017	9.434	10.320
	1000	11.795	12.187	11.683	12.154
100 000	600	5.181	5.498	5.265	5.498
	700	6.959	7.468	6.987	7.518
	800	9.551	12.141	9.523	11.721
	900	11.109	12.224	11.045	11.828
	1000	13.558	14.713	13.677	15.379

These results show that the two-phase approach, which is also using the idea of solving the LP-relaxation via an assignment procedure at the start, is outperforming the (one-phase) shortest augmenting path approach. Yet, the marginal reduction over the SMP(MP2PA)-code seems to indicate that the idea of solving the LP-relaxation efficiently is a dominant time saver. Note that for solving the assignment problem no data transformation is necessary since the matching data also defines the associated LP-relaxation/assignement problem and is used as input for the assignment code. Also for both problems — the assignment and the matching problem — we can use the same sparse subgraph in the two-phase approach.

Finally we present the results of experiments with the set of large matching problems considered by Grötschel and Holland. We first review the results cited in Grötschel and Holland's paper where they compare their implementation with the SMP-code from Burkard and Derigs on an IBM 4331. The running times include the time for input/output, i.e. for reading the data into main memory. (See Table 3.)

The data of these problems is stored on an external storage medium (disk) in a form which is suitable for the Grötschel-Holland code as a vector containing the upper triangular part of the symmetric cost matrix of the complete graph. All our codes require the input of the cost data in the form of a list. This required the

Table 3

Running times for basic SMP-code and the Grötschel-Holland code in IBM 4331 CPU-minutes

$\|V\|$	SMP-code	Grötschel-Holland approach
300	4:07	3:45
400	5:04	6:06
500	10:31	9:19
600	24:43	14:15
666	56:09	73:31
700	51:04	23:42
800	69:47	33:55
900	59:05	50:22
1000	66:35	61:09

transformation of the stored data. When discussing the computational results we also report the overall running time including input to be compatible with Grötschel and Holland's study. Since our data structure has about 50% redundancy (every edge is contained in two neighbourhoods) we can estimate that our input times are at least twice as long as Grötschel and Holland's.

In Table 4 we display the results for three of our matching codes mentioned above:

- the original SMP-code which was used in the Grötschel-Holland study,
- the SMP-code with the MP2PA-startroutine,
- the two-phase code TPHASE(8) (with $k = 8$).

In addition to the overall running time we display the time for input/output and the time for computing the optimal matching separately. No parts of the algorithm are hidden in the input time, the times given are spent on just reading the list structure representing the graph and the cost data from disk. When solving the problems from the other set, the data was generated every time and no input operations were necessary.

In Table 5 we display the time which the TPHASE(8)-code spends for solving the various phases/tasks during the overall procedure. Note that since we are solving the LP-relaxation/assignment problem via the two-phase approach the generation of the sparse subgraph is done as the first step.

Note that an entry of 0.0 in the "Reoptimization" column indicates that the problem was solved by applying the PREPHASE only, i.e. the optimal LP-relaxation (assignment) was integer-valued (a matching). Moreover, due to a special analysis of the (optimal) dual solution it is possible to reduce the number of superfluous outpricing operations and thus to speed up PHASE II significantly. The basic idea of this savings technique is given in Derigs and Metz (1986b) for the bipartite matching problem. The extension to the nonbipartite case is straightforward and can be found in Metz (1987).

Table 4

Running times on the Grötschel-Holland examples

n	Code	Total	Input	Optimization
300	SMP	56.28	14.80	41.48
	SMP(MP2PA)	16.10	14.20	1.90
	TPHASE(8)	16.00	14.61	1.39
400	SMP	86.83	26.40	60.43
	SMP(MP2PA)	28.15	25.38	2.77
	TPHASE(8)	28.13	26.03	2.10
500	SMP	136.61	41.14	95.47
	SMP(MP2PA)	44.99	39.71	5.28
	TPHASE(8)	44.05	40.71	3.34
600	SMP	365.24	59.25	305.99
	SMP(MP2PA)	71.66	56.98	14.68
	TPHASE(8)	63.76	58.88	4.88
666	SMP	773.75	73.30	700.45
	SMP(MP2PA)	240.32	70.17	170.15
	TPHASE(8)	87.23	72.59	14.64
700	SMP	721.86	80.77	641.09
	SMP(MP2PA)	85.33	77.59	7.74
	TPHASE(8)	85.77	79.88	5.89
800	SMP	1166.35	105.43	1060.92
	SMP(MP2PA)	148.47	101.40	47.07
	TPHASE(8)	113.66	104.22	9.44
900	SMP	757.37	134.83	622.54
	SMP(MP2PA)	146.24	128.55	17.69
	TPHASE(8)	141.41	132.13	9.28
1000	SMP	877.56	167.38	710.18
	SMP(MP2PA)	186.38	160.80	25.58
	TPHASE(8)	175.51	163.49	12.02

Table 5

Running times for TPHASE(8) spend for the different phases

n	Generation of sparse subgraph	Optimization of the		Reoptimization
		Assignment	Sparse problem	
300	0.702	0.699	0.0	0.0
400	1.125	0.985	0.0	0.0
500	1.651	1.504	0.143	0.037
600	2.283	2.070	0.493	0.050
666	3.162	2.190	6.673	2.609
700	3.025	2.902	0.0	0.0
800	3.827	3.691	1.794	0.100
900	4.786	3.844	0.609	0.080
1000	5.917	5.268	0.732	0.097

5. Concluding remarks

From the above we can draw the following information:

- The combinatorial two-phase approach outperforms the basic SMP-code and the Grötschel and Holland approach by more than an order of magnitude.
- The Grötschel–Holland examples seem to indicate that the real "computing time" of the algorithm may only be a fraction of the entire running time necessary to solve large problems due to the required input operations. Thus even further algorithmic improvements may not be able to reduce the overall solution time significantly.

The results on these artificial, i.e. randomly generated, problems show that in nearly all cases the optimal solution for the sparse problem induces an optimal matching and only a small amount of computing time is necessary to prove this optimality. Obviously it is easy to construct "bad" examples for the two-phase approach where the choice of the initial subgraph does not contain the optimal matching and which require to reoptimize, i.e. to add additional edges eventually several times. Even for arbitrary problems with a small cost-range the two-phase approach will lead to difficulties since either the initial subgraph has to be chosen quite dense or several enlargement steps of the working subgraph have to be allowed. Both choices will significantly increase the running time of any two-phase approach. So for instance solving the cardinality matching problem by the two-phase approach as a matching problem with two-valued edge cost may be infeasible.

Yet, from our experiments and the results given in Grötschel and Holland we feel that the SMP-based two-phase approach is less sensitive to subgraph size and the number of reoptimizations than the facet-generating two-phase approach. Since at any time of the procedure we can flip to optimization over the full graph the time for the SMP-code with the MP2PA-start is in a sense the worst we can expect. And also this code is a significant improvement over the basic SMP-code and other previously known matching codes.

References

R.E. Burkard and U. Derigs, "Assignment and matching problems: Solution methods with FORTRAN-programs," *Lecture Notes in Economics and Mathematical Systems, No. 184* (Springer, Berlin, 1980).

U. Derigs, "Postoptimal analysis for matching problems," *Methods of Operations Research* 49 (1980) 215–221.

U. Derigs, "Solving large-scale matching problems efficiently - A new primal matching approach," *Networks* 16 (1986) 1–16.

U. Derigs and A. Metz, "On the use of optimal fractional matchings for solving the (integer) matching problem," *Computing* 36 (1986a) 263–270.

U. Derigs and A. Metz, "An in-core/out-of-core method for solving large scale assignment problems," *Zeitschrift für Operations Research* 30 (1986b) A181–A195.

J. Edmonds, "Maximum matching and a polyhedron with 0, 1 vertices," *Journal of Research of the National Bureau Standards* 69B (1965) 125–130.

B. Gavish, P. Schweitzer & E. Shlifer, "The zero pivot phenomenon in transportation and assignment problems and its computational implications," *Mathematical Programming* 12 (1977) 226-240.

M. Grötschel and O. Holland, "Solving matching problems with linear programming," *Mathematical Programming* 33 (1985) 243-259.

A. Metz, "Postoptimale Analyse und neue primale Matching Algorithmen," Diploma Thesis, University of Cologne (Cologne, 1987).