Acta Cybernetica **14** (1999) 357-376.

On the Exact Solution of the Euclidean Three-Matching Problem

Gábor Magyar Mika Johnsson Olli Nevalainen *

Abstract

Three-Matching Problem (3MP) is an NP-complete graph problem which has applications in the field of inserting electronic components on a printed circuit board. In 3MP we want to partition a set of $n = 3l$ points into l disjoint subsets, each containing three points (triplets) so that the total cost of the triplets is minimal. We consider the problem where the cost c_{ijk} of a triplet is the sum of the lengths of the two shortest edges of the triangle (i, j, k); the reason for this assumption is the nature of the practical problems.

In this paper we discuss the optimal solution of 3MP. We give two different integer formulations and several lower bounds of the problem based on the Lagrangian relaxations of the integer programs. The different lower bounds are evaluated by empirical comparisons. We construct branch-and-bound procedures for solving 3MP by completing the best lower bound with appropriate branching operations. The resulting procedures are compared to our previous exact method and to general MIP solvers.

1 Introduction

The task in Three-Matching Problem (3MP) is to form l disjoint triplets from $n = 3l$ points and to minimize the total cost of these triplets, i.e., to connect the three points of each triplet by two line segments so that the total length of the line segments is minimal. The 3MP can be illustrated with an Euclidean problem instance, see Fig. 1.

The 3MP occurs in some industrial applications [3, 5, 7, 12]. In manual insertion of electronic components on a printed circuit board, the operations are arranged into close triplets to aid the worker's task. Furthermore, some flexible machines (e.g., General Surface Mounter) for automatic electronic component insertion have from three to eight insertion heads and operate in cycles comprising component pickups and insertions; the throughput of the machine can be improved by minimizing the length of the inter-board head movements. Component insertions are performed in two phases: in the first phase head nozzles pick up new components from the component feeders, and in the second phase the head moves to the actual insertion

*Turku Centre for Computer Science (TUCS) and Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland

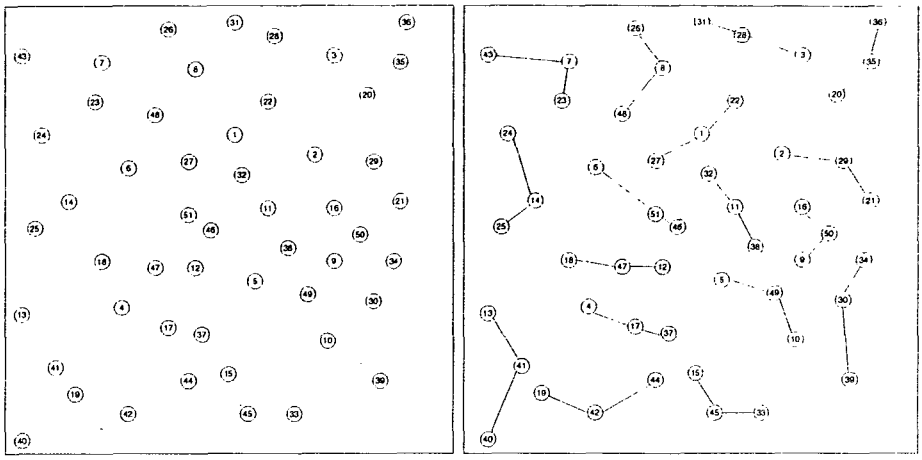

Figure 1: A sample 51-point 3MP problem instance ('eil51' from TSPLIB [15]) along with its optimal solution. The vertices are numbered as they appear in the problem file.

points to perform the actual insertions. Careful selection of the point triplets (in the case of a three-headed machine) is essential in minimizing the total printing time of the board. A similar problem is encountered in the scheduling of an automated assay analysis instrument (AutoDelphia).

3MP is closely related to the 3-dimensional assignment problem (3DA) [1, 3] which is NP-complete and a restricted version of our problem (in 3DA the points of the triplets are drawn from three disjoint subsets of the point set). Our previous studies and the connection to 3DA indicate that 3MP is NP-complete [6]. Additionally, 3MP is closely related to the p-median problem [2], in which we search p median points to which the remaining points are connected in such a way that the total sum of the connecting edges is minimal. However, the p-median problem does not restrict the number of allocated points to a median to be exactly n/p. The 3MP can therefore be viewed as a specialization of the p-median problem.

The 3MP can be solved heuristically by standard approaches, like local search heuristics using pairwise interchanges, simulated annealing, tabu search and genetic algorithm [6, 9, 10]. Several lower bounds can be given to the problem. By knowing the optimal solution to the problem we can evaluate the quality of the lower bound and the upper bound solutions of the heuristic approaches. Here we will discuss a number of design alternatives of B&B algorithms for 3MP and their trade-offs in this study.

The plan of the paper is as follows. Two different integer programming formulations for 3MP are given in the next section. In Section 3 we discuss briefly the branch-and-bound method and its main components. Section 4 describes four lower bounds to the problem and compares them empirically. The best lower bound is

completed with three different branching rules in Section 5. Test results comparing the branch-and-bound procedures and previous methods are discussed in Section 6. This section also compares our procedure to general MIP solver packages. Finally, the conclusions are drawn in Section 7.

2 Problem formulations

In this section, we give two integer programming formulations for 3MP. Firstly, the problem can be formulated by the following 0-1 program [6]:

Let d_{ij} represent the cost of the edge $j \to i$, V the set of vertices and x_{ij} the decision variable where

$$x_{ij} = \begin{cases} 1 & \text{if the edge } j \to i \text{ is present in the solution,} \\ 0 & \text{otherwise.} \end{cases}$$

The problem is to

$$\min z_{IP1} = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{i \in V} x_{ij} + \frac{1}{2} \sum_{k \in V} x_{jk} = 1 \qquad \forall j \in V \tag{2}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{3}$$

In the above formulation the edges are directed and the first index stands for the central point of the triplets. The first term in constraint (2) is equal to 1 if and only if the point j is not a central triplet point, whereas the second sum is equal to 1 if and only if the point j is a central point.

Our second formulation considers 3MP as a modification of the p-median problem (e.g., see [2]), where $p = n/3$ and every median has exactly two other points allocated to it. The decision variable x_{ii} is equal to 1 if and only if point i is a median vertex. Concerning the other variables,

$$x_{ij} = \begin{cases} 1 & \text{if point } j \text{ is allocated to median } i \text{ in the solution,} \\ 0 & \text{otherwise.} \end{cases}$$

The problem is to

$$\min z_{IP2} = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{4}$$

subject to

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{5}$$

$$\sum_{i \in V} x_{ii} = n/3 \tag{6}$$

$$\sum_{j \in V \& j \neq i} x_{ij} = 2x_{ii} \qquad \forall i \in V \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{8}$$

Equations (5) ensure that each point is allocated to a median. Equation (6) says that there are exactly $n/3$ medians. Equations (7) ensure that every point is allocated only to a median point and every median has exactly two other points allocated to itself. The above program is also valid for the 3MP if we have inequalities in (7).

When we describe the Lagrangian relaxations of the above integer programs in Section 4, we will refer to the first program as "IP1", the second one with signs "=" in (7) as "IP2a", and with signs "$\leq$" as "IP2b".

It must be noted that the p-median formulation does not hold for natural generalizations of the three-matching problem: for example, in the case of 4-matching when the objective is to find point groups of four points with a minimal total distance of the path crossing the four points, there is no median point.

3 Outline of the algorithm

The *branch-and-bound* (B&B) method (e.g., see [8]) is an intelligent enumerating procedure for finding the exact solution of a given combinatorial optimization problem. The method maintains a set of live leaves of an enumeration tree. The tree represents partitions of the feasible solutions in its nodes. The classes of the partitions are usually defined by storing some fixed variables or partial solutions in the nodes. The procedure begins with the root, which has no fixed variables; it represents all the feasible solutions of the optimization problem under consideration. The *branching* operation is used to partition the feasible solutions of a selected branching node into its descendant nodes. The partitioning is done by fixing some further variables in the descendant nodes in a systematic way depending on the structure of the problem. The effectiveness of the method comes from the *bounding* operation, where we calculate sharp lower bounds (in case of a minimization problem) of the object values of the feasible solutions represented in the nodes of the tree. We do not have to build up the whole enumeration tree in general, i.e., if a node does not contain any better solutions than a known one, then that node and the respective subtree can be discarded. A more precise description of the branch-and-bound procedure is as follows:

1. *(Initialization)* Let the set L of live leaves contain only one node, the root, which represents all the feasible solutions. Calculate the lower bound for the

root. Calculate a sub-optimal solution s^* with some *heuristics* if possible and store its value to z^*.

2. *(Iteration)* If L is empty, then terminate. Return s^* as the optimal solution and z^* as the optimal value.

3. *(Selection)* Select a branching node $Node$ from L with some leaf selection rule.

4. *(Branching)* Form the partition $\varphi(Node) = \{N_1, \ldots, N_k\}$ of the selected branching node.

5. *(Bounding)* Calculate the lower bounds $g(N_i)$ for all the new nodes, $i = 1, \ldots, k$. If a feasible solution arises during the bound calculations, then modify the current best solution s^* and its object value z^* when necessary.

6. *(Fathoming)* Update the set L of live leaves according to the explored node and the actual value of z^*:

   (a) add $N_1, \ldots, N_k$ to L,

   (b) delete $Node$ from L,

   (c) delete all $N \in L$ for which $g(N) \geq z^*$.

7. *(Next iteration)* Go to the next iteration (step 2).

We have two major alternatives to select a leaf at step 3. We can select the leaf with the minimal assigned lower bound. This way the set of live leaves usually grows very rapidly since we build the tree more or less level by level horizontally. A more practical approach selects the leaf for which the ratio of the assigned lower bound and the number of fixed variables is minimal. In this case we traverse the tree in a depth-first fashion. This action provides feasible solutions at an early phase of the processing and uses less memory. We will apply the second selection criterion in our branch-and-bound procedures.

The main components of a B&B procedure are the branching operation φ, the bounding function g, and the primal heuristics available for generating initial solutions. Furthermore, upper bound heuristics can also be applied to generate candidate solutions based on the partial solutions of the tree nodes. Next we consider all these components in detail and give a complete branch-and-bound procedure.

4 Lower bounds

In this section we discuss four lower bounds for the 3MP. The first bound is problem-specific, while the other three are based on the Lagrangian relaxations of IP1 and IP2a. The latter program provides two alternatives for the relaxation, either by relaxing the constraints (5) or (7).

The first lower bound ("LB1"), introduced in [6] (as bound "D") is applicable for the Euclidean case only:

1. For each point, calculate the distance to the closest point, store these distances together with the indices in the list S_1.

2. For each point, calculate the distance to the second closest point, store these distances together with the indices in the list S_2.

3. Sort lists S_1 and S_2 into an increasing order on the distances.

4. Omit duplicates from S_1 and S_2 (i.e., rule out the edge $b-a$ if $a-b$ has occurred previously), also rule out $a-c$ if $a-b$ and $b-c$ are already in the list and $a-c$ is the longest distance in the triplet $(a-b-c)$. The deletion is done first in S_2 and in S_1 only if S_1 itself requires the deletion (all deletion conditions are met in S_1). When selecting the distances from the sorted lists S_1 and S_2, we omit those distances that would connect a point to more than two other points.

5. If there are less than $2/3n$ values in S_1, move $2/3n - |S_1|$ elements with the shortest distances from S_2 to S_1. Alternatively, if $|S_1| > 2/3n$, delete at a maximum $|S_1| - 2/3n$ values from S_1. Deletion is allowed only if the points connected by the deleted edge are still connected after the deletion to some other point in S_1 by an edge. Deletions are started from the end of S_1 (the largest values first). The operation is implemented by maintaining a list $R = (r_1, \ldots, r_{|S_1|})$ where r_i is the degree of point i in S_1. So the deletion of the element (d_{ij}, i, j) is allowed only if $r_i > 1$ and $r_j > 1$. After a successful deletion, R is updated, $r_i = r_i - 1, r_j = r_j - 1$.

6. Sum the first $2/3n$ distances in S_1 to get the lower bound.

The Lagrangian relaxation (see e.g. [4, 14]) is a general way for obtaining high quality lower bounds for hard combinatorial optimization problems. By attaching Lagrangian multipliers to some of the constraints of the original problem and relaxing these constraints into the objective function, we get a Lagrangian relaxation, which is easier to solve than the original problem. Maximizing the optimal values of the Lagrangian relaxation for λ, we obtain the Lagrangian dual program. The optimal value of the Lagrangian dual is a valid lower bound (in the case of minimization) and in the optimal case it can reach the optimal value of the linear programming relaxation of the original problem. The main advantage of the Lagrangian relaxation is that the solution process is much faster than solving the LP relaxation. The maximization of the value of the Lagrangian dual is often done by the subgradient optimization method, in which we update the Lagrangian multipliers in a systematic way to achieve the best lower bound. The Lagrangian relaxation technique has been applied successfully to many hard combinatorial optimization problems, see for example, [1, 2, 4].

Next we describe three Lagrangian lower bounds and the subgradient optimization procedure for maximizing the bounds. Our first bound is based on the Lagrangian relaxation of IP1 [6]. The latter two relaxations were introduced in [2] for the general p-median problem. Here we recall them with such modifications that make them applicable to the 3MP.

In the case of the relaxation for the problem formulation of IP1, we relax the constraints (2) into the objective function by introducing the Lagrangian multipliers $\lambda_j (j \in V)$ and get the following Lagrangian relaxation:

$$\begin{aligned} \min z_{D1} &= \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} + \sum_{j \in V} \lambda_j \left(\sum_{i \in V} x_{ij} + \frac{1}{2} \sum_{k \in V} x_{jk} - 1 \right) = \\ &= \sum_{i \in V} \sum_{j \in V} \left(d_{ij} + \lambda_j + \frac{1}{2} \lambda_i \right) x_{ij} - \sum_{j \in V} \lambda_j \end{aligned} \tag{9}$$

subject to

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{10}$$

This program is easy to solve. The solution becomes straightforward after examining the signs of the terms $D_{ij} = d_{ij} + \lambda_j + 0.5\lambda_i$ since the second sum of (9) is constant with respect to x. Hence, the optimal solution of z_{D1} is:

$$x_{ij}^* = \begin{cases} 0 & \text{if } D_{ij} > 0, \\ 1 & \text{if } D_{ij} \leq 0. \end{cases} \tag{11}$$

Now we describe two Lagrangian relaxations for the program IP2a. Let K_1 denote the set of vertices that have been previously fixed to be medians and K_0 the vertices that have been fixed to be non-medians by the branching rules of the B&B procedure.

As a second relaxation, we relax the constraints (7) by introducing the multipliers $\lambda_i (i \in V)$. The resulting program allows us to allocate the points to non-medians and, in addition, a point may have an arbitrary number of other points allocated to itself. We get the following Lagrangian relaxation:

$$\begin{aligned} \min z_{D2} &= \sum_{i \in V - K_0} \sum_{j \in V - K_1} d_{ij} x_{ij} + \sum_{i \in V - K_1} \lambda_i \left(\sum_{j \in V \& j \neq i} x_{ij} - 2x_{ii} \right) = \\ &= \sum_{i \in V - K_0} \sum_{j \in V - K_1 \& j \neq i} (d_{ij} + \lambda_i) x_{ij} + \sum_{i \in V - K_0} (d_{ii} - 2\lambda_i) x_{ii} \end{aligned} \tag{12}$$

subject to

$$\sum_{i \in V-K_0-K_1} x_{ij} = 1 \qquad \forall j \in V - K_1 \tag{13}$$

$$\sum_{i \in V-K_0-K_1} x_{ii} = n/3 - |K_1| \tag{14}$$

$$\lambda_i = 0 \qquad \forall i \in K_1 \tag{15}$$

$$x_{ii} = 0 \qquad \forall i \in K_0 \tag{16}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{17}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in V, j \in V \tag{18}$$

To solve this program, we define α_j as the minimum cost of allocation vertex j other than to itself, i.e., the minimal arising cost if vertex j is not a median:

$$\alpha_j = \min_{i \in V-K_0 \& i \neq j} (d_{ij} + \lambda_i) \qquad \forall j \in V - K_1 \tag{19}$$

The Lagrangian relaxation (equations (12)-(18)) then can be changed for the following problem which has the same optimal value as the Lagrangian relaxation. The basis of this substitution is that the minimal value of $\sum_{i \in V-K_0, i \neq j} (d_{ij} + \lambda_i) x_{ij}$ is equal to α_j if $j \in V - K_1$, by (13).

$$\min z_{D2} = \sum_{j \in V-K_1} \alpha_j + \sum_{j \in K_1} d_{jj} + \sum_{j \in V-K_0-K_1} (d_{jj} - 2\lambda_j - \alpha_j) x_{jj} \tag{20}$$

subject to

$$\sum_{i \in V-K_0-K_1} x_{ii} = n/3 - |K_1| \tag{21}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{22}$$

$$x_{ii} \in \{0,1\} \qquad \forall i \in V - K_0 \tag{23}$$

The optimal solution of this program is found by setting x_{ii} to 1 for $i \in K_1$ and the remaining $n/3 - |K_1|$ x_{ii} with the smallest $(d_{ii} - 2\lambda_i - \alpha_i)$ $(i \in V - K_0 - K_1)$ to 1, while all the other x_{ii} are set to zero. If the values of the variables x_{ii} are denoted by x^*_{ii} the optimal solution of the above program, then the other variables (x^*_{ij}) are calculated as follows:

$$x^*_{ij} = \begin{cases} 1 & \text{if } x^*_{jj} = 0 \text{ and } i \text{ corresponds to the minimum} \\ & \text{for } \alpha_j \text{ in (19) } (i \neq j), \\ 0 & \text{otherwise } (i \neq j). \end{cases} \tag{24}$$

Thirdly, we relax the constraints (5) of IP2a by introducing the Lagrangian multipliers $\lambda_j (j \in V)$ and get the following Lagrangian relaxation:

$$\begin{aligned} \min z_{D3} &= \sum_{i \in V-K_0} \sum_{j \in V-K_1} d_{ij} x_{ij} - \sum_{j \in V} \lambda_j \left(\sum_{i \in V} x_{ij} - 1 \right) = \\ &= \sum_{i \in V-K_0} \sum_{j \in V-K_1} (d_{ij} - \lambda_j) x_{ij} + \sum_{j \in K_1} (d_{jj} - \lambda_j) + \sum_{j \in V} \lambda_j \end{aligned} \tag{25}$$

subject to

$$\sum_{i \in V-K_0-K_1} x_{ii} = n/3 - |K_1| \tag{26}$$

$$\sum_{j \in V-K_1 \& j \neq i} x_{ij} = 2x_{ii} \qquad \forall i \in V \tag{27}$$

$$x_{ii} = 0 \qquad \forall i \in K_0 \tag{28}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{29}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in V, j \in V \tag{30}$$

The relaxation disregards the constraint that every point must be allocated to a median. Therefore, every selected median point will still have exactly two other points allocated to it, but a point can be allocated to several medians.

To solve this program, let us consider the effect of specifying that k is a median vertex. This implies the settings $x_{kk} = 1$ and $x_{kj} = 1$ where j corresponds to the indices of the smallest or second smallest value of $(d_{kj} - \lambda_j)$, $j \in V - K_1 - \{k\}$. This is true, because the constraints (27) ensure that every selected median will have the two points with the smallest $(d_{kj} - \lambda_j)$ allocated to itself. Because the constraints (5) have been relaxed, a point can be allocated to several medians. We denote by α_k the arising cost when deciding that vertex k will be a median:

$$\alpha_k = (d_{kk} - \lambda_k) + \min_{j \in V-K_1 \& j \neq k} (d_{kj} - \lambda_k) + \operatorname{2ndmin}_{j \in V-K_1 \& j \neq k} (d_{kj} - \lambda_k), \forall k \in V - K_0 \tag{31}$$

where "2ndmin" denotes the second smallest value of the expression.

The solution of the Lagrangian relaxation of (25)-(30) now can be obtained by solving the following problem which has the same optimal value:

$$\min z_{D3} = \sum_{i \in V-K_0} \alpha_i x_{ii} + \sum_{j \in V} \lambda_j \tag{32}$$

subject to

$$\sum_{i \in V - K_0 - K_1} x_{ii} = n/3 - |K_1| \tag{33}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{34}$$

$$x_{ii} \in \{0,1\} \qquad \forall i \in V - K_0 \tag{35}$$

The optimal solution of this program is found by selecting the needed $n/3 - |K_1|$ medians as follows: set x_{ii} to 1 for the indices i ($i \in V - K_0 - K_1$) which give the smallest α_i values. The other x_{ii} values (except $i \in K_1$) are set to 0. Denoting the optimal assignment of the diagonal elements by x^*_{ii}, the remaining variables (x^*_{ij}) are allocated as follows:

$$x^*_{ij} = \begin{cases} 1 & \text{if } x^*_{ii} = 0 \text{ and } j \text{ corresponds to the smallest or the} \\ & \text{second smallest index for } \alpha_i \text{ in (31) } (i \neq j), \\ 0 & \text{otherwise } (i \neq j). \end{cases} \tag{36}$$

The problem reduction by calculating penalties is a common technique in Lagrangian relaxations. Suppose that we have solved the Lagrangian relaxation to optimality. We can then estimate the increase in the lower bound which would result from forcing a variable to either 0 or 1. If the lower bound resulting from imposing some condition on a variable is above some known upper bound to the problem, then that condition cannot be satisfied in the optimal solution of the original problem. This means that we can fix the value of the given variable as it is in the optimal solution of the Lagrangian relaxation. For example, if a variable x^*_{ij} is 0 in the optimal solution of the Lagrangian relaxation, and the penalty for forcing x^*_{ij} to be 1 results in exceeding the known upper bound, then the value of x^*_{ij} can be fixed to 0 and the size of the problem can be reduced. For the third relaxation, we include the calculation of the penalties, where we have four cases for $x^*_{ij} = 0$ and two cases for $x^*_{ij} = 1$.

Finally, we describe the method that is applied for maximizing the optimal values of the Lagrangian dual programs. We will denote the three described Lagrangian relaxations by "LR1", "LR2" and "LR3" and "LR3P" (LR3 with penalties). In the above relaxations, the Lagrangian multipliers are unconstrained in sign, because the relaxed constraints are equalities in every case. We apply the following subgradient optimization procedure in order to maximize the value of the Lagrangian dual programs:

1. Determine an upper bound z_{UB} to the problem. This can be done by any of our previous heuristic methods, see for exapmle, [6, 9, 10]. Set z^*_L to $-\infty$, which will denote the maximal lower bound.

2. Initialize the Lagrangian multipliers to $\lambda_j = 0$, $j \in V$.

3. Solve the Lagrangian relaxation with the actual set of Lagrangian multipliers. Let z^*_D denote the optimal value of the relaxation, x^*_{ij} the optimal solution and M^* the associated median set (in the case of LR2 and LR3).

4. Set $z_L^* = \max(z_D^*, z_L^*)$.
5. Terminate if $z_{UB} \leq z_L^*$, which means that the maximal lower bound is found. Otherwise go to step 6.
6. Perform the penalty tests outlined above (case LR3P).
7. Calculate the subgradient vector S:

$$S_j = \sum_{i \in V} x_{ij}^* + \frac{1}{2} \sum_{k \in V} x_{jk}^* - 1 \qquad \forall j \in V \text{for LR1} \tag{37}$$

$$S_i = \sum_{j \in V \& j \neq i} x_{ij}^* - 2x_{ii}^* \qquad \forall i \in V \text{for LR2} \tag{38}$$

$$S_j = 1 - \sum_{i \in M^*} x_{ij}^* \qquad \forall j \in V \text{for LR3} \tag{39}$$

8. If $S_j = 0$ for all $j \in V$, then terminate; the optimal solution of the original problem is found. Otherwise go to step 9.
9. Calculate the step size T for updating the multipliers by

$$T = \frac{\pi(z_{UB} - z_D^*)}{\sum_{i \in V} S_i^2} \tag{40}$$

where π is a constant $(0 < \pi \leq 2)$ controlling the step size of the procedure.

10. Update the multipliers by

$$\lambda_j = \lambda_j + TS_j, \qquad \forall j \in V \tag{41}$$

11. Go to step 3 for the next subgradient iteration unless some termination criteria (see below) is satisfied. Else stop and return z_L^* as the maximal lower bound.

We set the value of π initially to 2. After that it is halved after every 30 iterations if the value of the best lower bound has not been improving. The procedure terminates after reaching a predefined total number of iterations which is set to 200.

Now we are in the position to compare the lower bounds. Table 1 shows a summary of the results of practical tests with 20 Euclidean problem instances[1]. It should be noted that all the above bounds except the first one (LB1) can deal with

[1]The test problems are available from http://www.cs.utu.fi/research/projects/3mp/

Prob	Pts	LR3P	LR3	LR1	LR2	LB1	max LB	OPT
f21	21	0,00	0,03	8,31	8,31	7,52	159,73	159,73
f27	27	1,22	1,22	7,88	7,91	10,45	7913,53	8011,08
f33	33	0,00	0,00	7,31	7,32	8,31	255,69	255,69
f39	39	10,63	10,63	15,79	15,79	14,31	252,31	282,31
39a	39	1,56	1,56	9,98	9,98	7,55	771,74	783,96
39b	39	7,47	7,47	17,03	17,06	16,76	764,45	826,14
39c	39	0,54	0,54	9,81	9,84	11,44	954,21	959,38
39d	39	1,65	1,65	10,41	10,44	10,34	768,33	781,22
39e	39	6,25	6,25	13,00	13,01	16,11	817,82	872,35
42a	42	6,14	6,14	15,14	15,16	15,77	890,83	949,09
42b	42	5,41	5,41	15,43	15,48	15,28	814,00	860,59
45a	45	0,55	0,55	10,92	10,94	10,52	1007,60	1013,14
45b	45	4,20	4,20	20,34	20,36	22,62	944,18	985,60
48a	48	0,00	0,00	5,12	5,13	4,84	996,61	996,61
48b	48	0,00	0,00	15,52	15,53	11,66	967,83	967,83
51a	51	1,78	1,78	14,19	14,20	13,26	966,01	983,55
51b	51	3,01	3,01	11,07	11,09	9,17	973,57	1003,82
eil51	51	2,36	2,36	8,33	8,33	8,66	259,34	265,61
f99	99	0,89	0,89	5,99	5,99	6,88	382,78	386,23
rat99	99	1,07	1,07	8,88	8,89	9,60	743,48	751,53
Average		2,7367	2,7382	11,5233	11,5372	11,5542		

Table 1: Test results for different lower bounds and the effect of the penalties for the bound of LR3. The values show the difference from the optimal value in percents.

any kind of distance matrix. The problem instances "rat99" and "eil51" are from the TSPLIB [15] and the instances "f21", "f27", "f33", "f39", "f99" are prefixes of the files "eil51", "krob150", "rat99", "rat783" and "eil101" of the TSPLIB, respectively. The other 13 files are generated by picking the co-ordinates of the points randomly and independently within the range [1,300]. The results of Table 1 show the difference from the optimum. Column 8 shows the value of the maximal lower bound of the five bounds and the last column shows the optimal value, which we calculated with the B&B procedure. The averages of the differences from the optimum are indicated in the last row.

The results indicate that LR3 outperforms the other bounds. Furthermore, the penalties still improve the bound in one case (f21). The average difference between the bound LR3 (LR3P) and the optimum is 2.7367% for the test problems. Some problems (e.g., f39, 39b and 39e) are harder than the others because the bound is farther from the optimum. On the other hand, the bound can also be very close to the optimum. In some cases it even reaches the optimum, for exapmle, f21, f33, 48a and 48b.

The results of LR1 and LR2 are practically identical and LB1 also seems to be as good as these two bounds. The difference between LR3 and the three other bounds is relatively large. Therefore, the application of LR3 is justified in the B&B procedure, as the quality of the lower bound is essential. However, the calculation

of the bound LB1 is much faster than of the other three, which require many hundreds of iteration steps of the subgradient method. Therefore, we will examine the application of this bound in the B&B procedure.

Finally, the results of Table 1 are also in line with the observations of [2], i.e., the bound LR3 clearly outperforms LR2.

5 Branching rules

Next we equip LR3P with appropriate branching rules to build up a complete branch-and-bound procedure for the 3MP. We describe three different branching rules. The first one was introduced in [2] for LR3. The other two can be considered as enhancements or extensions of the first one. The branching rules choose variables to the sets of K_0 (non-medians) and K_1 (medians) (cf. Section 4) systematically. A variable is referred as a free variable unless the branching rules or some other operations, for example the application of the penalties, have fixed its value.

Rule 1. The first branching rule [2] selects the variable x_{jj} corresponding to

$$\alpha_j = \min_{i \in M^* - K_1} \alpha_i \tag{42}$$

and sets x_{jj} either to 0 (non-median, $K_0' = K_0 + \{j\}$) or to 1 (median,$K_1' = K_1 + \{j\}$) in the descendant nodes. This rule selects the median point from the selected free medians of the Lagrangian relaxation and gives a binary enumeration tree.

Rule 2. This rule performs a 3-ary branching based on a free variable x_{jk}, where j is determined by (42) as above. The vertex k corresponds to $\min d_{jk}$ among the free variables x_{jk} (i.e., it represents the smallest possible allocation cost of a point to j). In one branch, we perform the partitioning by fixing x_{jj} to 0 ($K_0' = K_0 + \{j\}$). In the other two branches, we set j as a median by fixing $x_{jj} = 1$ ($K_1' = K_1 + \{j\}$). The latter two branches differ in the value of the variable x_{jk}, namely we fix $x_{jk} = 1$ in one of them (this also yields that k becomes a non-median), and $x_{jk} = 0$ in the other. This rule is an extension of rule 1 in the sense that in addition to deciding whether point j is a median or not, we also decide an allocated point k in the case when j was selected as a median.

Rule 3. This rule is a modification of rule 2. We select now the branching variable x_{jk} corresponding to $\min d_{jk}$, where $j \in V - K_0 - K_1, k \in V - K_1, k \neq j$ and x_{jk} is free. The difference from the previous rule is that the candidate for the median point (j) is not determined by (42), but it is selected according to the minimal value of d_{jk}, where j can be any of the possible candidates for a median. The branching is similar to rule 2, this again yields a 3-ary tree.

The above branching rules are exhausted when we have fixed the required number of medians, i.e., $|K_1| = n/3$ (or $|K_0| = 2n/3$). It seems to be problematic that in the p-median problem the solution resulting from the relaxation LR3 is restricted to find the $n/3$ median points. In the general p-median problem, the

remaining points are allocated to the closest medians and a median can have an arbitrary number of other points allocated to it. The solution of 3MP, however, requires that each median has exactly two other points allocated. This difficulty can be overcome by applying the Hungarian method (e.g., see [13]) as follows: Let us create an assignment problem where we put the median points in duplicates into the first set (i.e., every selected median point will appear twice). The other set includes the non-median points. Performing the Hungarian method for the assignment of the elements of the two sets results for each median point a matching to exactly two non-median points, where the allocated points are different for each pair of median points. With this additional procedure, the solution of 3MP is complete after solving the appropriate restricted version of the general p-median problem. For this reason, it is enough to select the $n/3$ median points optimally (central points of the triplets), and the exact solution of the 3MP can be derived from that.

In order to find tight upper bounds during the B&B procedure, we perform the Hungarian method also on the median set M^* which is associated with the best Lagrangian lower bound from the subgradient procedure. This way the size of the tree can be reduced since the better upper bounds enable us to fathom some more unnecessary leaves, see step 6 of the B&B procedure in Section 3.

It was noted above that the calculation of a high quality lower bound is achieved by 200 iterations of the subgradient method. This is applied at the root node. We perform only 30 iterations at other levels of the tree. The Lagrangian multipliers of the tree nodes are initialized to their best values in the parent nodes; this common practice enables a smaller number of iterations at the lower levels of the tree. The value of π is initialized to 1 in the tree nodes, and it is halved after every fifth iteration.

When adding a variable to the set K_0, all the variables in its row are also fixed to 0, as it cannot have any points allocated. If a variable is added to K_1, the variables in its column (except the diagonal) are fixed to 0, as it cannot be allocated to other points. We apply some additional tests on the fixed variables, and, if it is possible, we still fix some more variables. These additional tests take place after the penalty tests and they work recursively until no new variable has been fixed.

The following tests are performed:

1. If all three variables have been assigned to the median (the median and the two allocated points), then all the other free variables of the row of the median can be fixed to 0.

2. If a row of a median contains only the required number of free variables (i.e., 1 or 2), then those variables are automatically allocated to the median; furthermore, the appropriate columns are included in the set of K_0. The other variables in the row and column of the automatically allocated variables are fixed to 0.

3. If a row has less than three free variables and no variable has been fixed to 1, then it is impossible for it to be a row of a median. Therefore, the

		Time (sec)			Nodes		
Problem	Pts	P1	P2	P3	P1	P2	P3
f27	27	17	7	12	91	19	73
f39	39	N/A	76803	31903	N/A	N/A	323599
39a	39	24	27	16	127	169	73
39b	39	N/A	16935	4946	N/A	143098	29068
39c	39	28	20	9	131	115	13
39d	39	44	29	24	195	208	109
39e	39	9490	5034	741	46847	46789	3718
42a	42	14359	7093	2348	65345	52468	12310
42b	42	51916	17670	9847	120969	142495	59497
45a	45	19	9	11	59	13	25
45b	45	5720	5226	5573	22121	33475	33484
51a	51	558	58	105	2169	259	439
51b	51	6832	2962	554	29471	17557	2296
eil51	51	4216	1010	284	15243	4984	1141
f99	99	N/A	3667	4055	N/A	10546	7228
rat99	99	N/A	9692	3444	N/A	23731	7012

Table 2: A comparison of the branching rules

corresponding vertex is included in K_0 and all the elements of the row are fixed to 0.

4. If there is only one free variable in a column, then it is fixed to 1. The necessary additional operations are also performed, i.e., the index corresponding to the row of the fixed variable is included in K_1 if it has not yet been included.

Finally, some feasibility tests are also performed after each subgradient iteration, i.e., if too many medians or non-medians have been fixed by the deterministic rules, or all the elements of a column have been fixed to 0, or there is no free point to allocate to a previously identified median, then the corresponding tree node will be fathomed.

Now we have the components for a complete branch-and-bound procedure. In the light of the previous experiments on the quality of the lower bounds, we will apply the bound which is based on our third relaxation (LR3). Next we examine the efficiency of the branching rules, see Table 2. All three procedures apply the penalties and the additional variable tests when calculating the lower bound. They differ only in the branching rules they apply, and we denote them by "P1", "P2" and "P3", respectively. The running times[2] (in seconds) and the number of examined nodes are indicated in Table 2 for the selected problem instances.

The results show that the second and third branching rule have significantly better performance than the first rule. Furthermore, the third rule is better than

[2]All computational tests were performed on a 133MHz Pentium PC.

		Time (sec)			Nodes		
Problem	Pts	P3	P3-	P3- -	P3	P3-	P3- -
f21	21	1	2	19	1	16	40
f27	27	12	26	187	73	244	436
f33	33	1	1	3	1	1	1
39a	39	16	28	298	73	169	346
39c	39	9	29	171	13	181	139
39d	39	24	32	275	109	160	268
39e	39	741	1617	26072	3718	10867	17242
45a	45	11	15	205	25	43	130
51a	51	105	675	7515	439	4096	5284
51b	51	554	4086	32615	2296	22360	23635
eil51	51	284	851	11843	1141	4663	9352

Table 3: The effect of the penalty tests and the additional variable tests

the second one, and therefore, the application of that rule is desirable in an efficient branch-and-bound procedure. The average ratio of the running times of the procedures P1 and P3 was roughly five; for P2 and P3 the corresponding value was two. On average, the procedure P1 generated ca. five times more nodes than P3, while the average ratio of the number of nodes for P2 and P3 was about three. The average processing time per a node was approximately the same for all three procedures.

Concerning the efficiency of the penalties and the additional variable tests, Table 3 presents a comparison which is based on some of the easier problem instances. In the light of the experiments with the branching rules (see Table 2), the third branching rule is applied in all three procedures. The procedures differ only in the way they apply the penalties and the variable tests: "P3" denotes the variant which applies both the penalties and the additional tests (this variant was also included in the previous tests), "P3-" denotes the variant where the penalties are applied but the additional variable tests are not, "P3- -" denotes the variant without the penalties and without the additional variable tests. The table indicates the running times in seconds and the number of examined nodes for the selected problems.

The results of Table 3 clearly show the advantages of applying both the penalties and the additional variable tests. The penalties generally enable to fix a large portion of the variables and have two main advantages. Firstly, by fixing many of the variables, the calculation of the lower bound becomes much faster. Secondly, the performance of the branch-and-bound procedure based on the lower bound equipped with the penalty tests is much better, since eliminating many of the free variables reduces the size of the enumeration tree.

The average ratio of the running times of the procedure without the penalties (P3- -) and with the penalties (P3-) was roughly ten, while the average ratio of the number of nodes was two for the same procedures. Concerning the effect of the additional variable tests, the average ratio of the running times of the procedure

without these tests (P3-) and with these tests (P3) was roughly three, whereas the average ratio of the number of nodes was six for the same procedures. The average processing time of one tree node was approximately 3.5 seconds for P3, 5.7 for P3-, and 1.0 for P3- -. These values correspond to the observation that the application of the penalties demands more processing time. The number of nodes is, however, much smaller when penalties are applied. The additional variable tests speed up the calculation, and their application also yields a smaller number of tree nodes on average. To summarize, the experiments confirm that the application of both the penalties and the additional variable tests is advantageous.

6 Comparisons to other approaches

In this section, we compare the performance of P3 with our previous B&B variant ("P-LB1") [6] and with general MIP solver packages "OSLMIP"[3] and "lp-solve"[4]. The P-LB1 is based on the quickly computable lower bound LB1. The branching is done by selecting the free variable with the smallest cost and fixing its value either to 0 or to 1 in the two descendant nodes. The P-LB1 also performs some variable manipulating procedures, which can fix certain additional variables on the basis of the previous branching decisions.

Table 4 shows the running times for the different procedures in seconds. The better solver (OSLMIP) was performed on all the three integer programs, i.e., on IP1, IP2a and IP2b, while the other (lp-solve) is performed on IP1 only.

The results of lp-solve with IP2a and IP2b are not included in Table 4, because they were much worse than with the formulation IP1. We could obtain results only for the first two problems. These results were roughly 6-10 times worse than the ones with IP1. For the other problems, we interrupted the execution of the procedures after 8-10 hours without termination. These observations are especially interesting if we compare them with the results of the other solver, which showed that the latter two formulations are much more promising to solve.

The results of Table 4 show that the procedure P3 outperforms the procedure P-LB1 due to the sharper bounding function. The calculation of the specific bound of P-LB1 is much faster than the bound of P3, but the improvement in the quality of the bound yields a much smaller tree size and total running time.

Concerning the comparison with the general MIP solvers, the results are diverse. The highly optimized machine code of the first solver (OSLMIP) gives better results than our procedure, while the general package (lp-solve) was much worse than our method. We think that differences in the running time between our procedure and the better solver may lay mainly in technical details, as we used general techniques and high-level programming languages for the implementation. The ratios between the running times of P3 and OSLMIP show that the difference does not grow with

[3]The package called OSLMIP is developed by IBM and it is one of the market leaders with respect to the performance. We downloaded the 60-day trial version from http://ism.boulder.ibm.com/es/oslv2/startme.htm

[4]The package lp-solve can be downloaded from ftp://ftp.ics.ele.tue.nl/pub/lp_solve/

Problem	Pts	OSL-IP1	OSL-IP2a	OSL-IP2b	P-LB1	P3	LP-IP1
f21	21	10	7	4	8	1	34
f27	27	19	8	12	59	12	386
f33	33	10	4	4	98	1	181
f39	39	8114	620	814	122689	31903	N/A
39a	39	76	16	43	67	16	98
39b	39	7642	644	206	74035	4946	N/A
39c	39	71	13	15	2541	9	21452
39d	39	32	14	27	1000	24	2369
39e	39	1220	402	160	59605	741	20435
42a	42	3670	584	406	31272	2348	231900
42b	42	5787	452	543	N/A	9847	N/A
45a	45	88	16	25	N/A	11	N/A
45b	45	423	107	62	N/A	5573	N/A
48a	48	9	9	7	145	1	200
48b	48	39	8	8	N/A	3	N/A
51a	51	130	23	31	186560	105	N/A
51b	51	841	302	88	N/A	554	N/A
eil51	51	777	613	133	N/A	284	N/A
f99	99	2889	718	1312	N/A	4055	N/A
rat99	99	N/A	3545	2100	N/A	3444	N/A

Table 4: A comparison of some methods for finding the exact solution of the 3MP

the problem size but stays at a constant level. However, for some hard problems this difference can increase, see problems f39, 39b, 42b, 45b.

Experiments with problem instances with different cost matrix show a rather different trend [11]. It turned out that matrices with the triangle inequality property are hard for our procedure, while on totally random cost matrices our procedure is superior to the other approaches.

Comparing the efficiency of the MIP solvers with respect to the applied integer program, as stated above, it is important which formulation is used as an input. The results of OSLMIP show that the modified p-median formulation is much better. Moreover, IP2b seems to be slightly better than IP2a.

7 Conclusions

We have considered the exact solution of an NP-complete graph problem, the three-matching problem. We introduced a problem-specific lower bound and a Lagrangian lower bound based on an integer formulation to the problem. Based on the connection to the p-median problem [2], two further Lagrangian relaxations were given to the problem. We gave a solution of the 3MP by modifying an algorithm for the p-median problem.

Our empirical results showed that the lower bounds were quite different, and special care must be taken when selecting the constraints to be relaxed to the

objective function. It was also shown that two seemingly different approaches gave the same lower bounds (LR1 and LR2). The application of the penalty tests speeds up the calculation of the lower bound considerably because it eliminates a large portion of the free variables.

We constructed several complete B&B procedures by introducing different branching rules. Our modification of the branching rule of [2] gave the best performance. Furthermore, we incorporated additional variable tests in order to enable to fix more free variables. The efficiency of the branching rules were examined and the application of the penalties and the additional variable tests were justified by empirical tests.

We compared our B&B procedure with our previous algorithm, which uses a quickly computable but worse lower bound, and the results clearly showed that the increase in the quality of the bound was essential and the procedure with the better bound outperformed the previous approach. Furthermore, we compared our procedure with two general optimization packages, and the results were diverse. The highly optimized machine code of a commercial MIP solver had better running times than our procedure, but the ratios of the two running times seemed to remain constant. The other solver, which was written in C, had a worse performance than our procedure.

It was also shown that it was very important which integer program was used in the MIP solvers. There is a significant difference between the formulations IP1 and IP2, but even the small difference between the formulations of IP2a and IP2b can yield great differences in the performance of the solvers applied to the two programs.

References

[1] E. Balas and M.J. Saltzman: An algorithm for the three-index assignment problem, *Operations Research 29 (1991)*, 150-161.

[2] N. Christofides and J.E. Beasley: A tree search algorithm for the p-median problem, *European Journal of Operational Research* 10 (1982), p. 196-204.

[3] Y. Crama, A.W.J. Kolen, A.G. Oerlemans and F.C.R. Spieksma: *Production Planning in Automated Manufacturing*, Springer-Verlag, 1994.

[4] M.L. Fisher: The Lagrangian relaxation method for solving integer programming problems, *Man. Sci. 27 (1981)*, 1-18.

[5] M. Johnsson, T. Leipälä: Determining the manual setting order of components on PC-boards, *Journal of Manufacturing Systems*, Vol. 15 No. 3 (1996), 155-163.

[6] M. Johnsson, G. Magyar, O. Nevalainen: On the Euclidean 3-Matching Problem, *Nordic Journal of Computing* 5(1998), p. 143-171.

[7] P. J. M. van Laarhoven and W. H. m. Zijm: Production Preparation and Numerical Control in PCB Assembly, *The International Journal of Flexible Manufacturing Systems*, 5 (1993), p. 187-207.

[8] E.L. Lawler and D.E. Wood, "Branch-and-Bound Methods: A Survey", *Operations Research 14 (1966)*, 699-719.

[9] G. Magyar, M.Johnsson, O. Nevalainen: Genetic algorithm approach for the three-matching problem, *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*, p. 109-122, Aug. 1997. (ftp://ftp.uwasa.fi/cs/3NWGA/Magyar.ps.Z)

[10] G. Magyar, M.Johnsson, O. Nevalainen: An adaptive hybrid GA for the 3-matching problem, *Turku Centre for Computer Science (TUCS) Technical Report 166*, March, 1998.

[11] G. Magyar, M.Johnsson, O. Nevalainen: On the exact solution of the three-matching problem, *Turku Centre for Computer Science (TUCS) Technical Report 199*, October, 1998.

[12] K. Palletvuori, P. Luostarinen, K. Muurinen and O. Nevalainen: "On the scheduling of a multipurpose laboratory analysis instrument", In *9th Euromicro Workshop on Realtime Systems 97*, 1997.

[13] C.H. Papadimitrou and K. Steiglitz: *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, New Jersey, 1982.

[14] *Modern Heuristic Techniques for Combinatorial Problems*, edited by C.R. Reeves. McGraw-Hill, 1995.

[15] G. Reinelt: TSPLIB - A Traveling Salesman Problem Library, *ORSA Journal on Computing 3 (1991)*, 376-384.