Pivot-and-Fix; A Mixed Integer Programming Primal Heuristic

Mahdi Namazifar*, Robin Lougee-Heimer **, and John Forrest ***

October 2009

Abstract. Pivot-and-Fix, a new primal heuristic for finding feasible solutions for mixed integer programming (MIP) problems is presented in this paper. Pivot-and-Fixtries to explore potentially promising extreme points of the polyhedron of the problem and by forming smaller serach trees looks for integer feasible solutions. Computational results show that this heuristic could help MIP solvers to perform faster.

Key words: Pivot-and-Fix, primal heuristics, mixed integer program, extreme point.

1 Introduction

Mixed integer programming (MIP) is a class of mathematical programming which deals with optimization problems with linear objective function and constraints, and a mix of conitinuous and integer variables. The following formulation is the general form of MIP problems:

$$(P) \qquad \begin{aligned} \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^p} \; & cx + fy \\ \text{s.t. } & Ax + Gy \geq b \\ & x \geq 0, y \in \mathbb{Z}^p_+. \end{aligned}$$

There is an enormous number of fields that benefit from MIP including manufacturing, transportation, finance, etc. The general and most frequently used approach to solve this class of problems is tree search. Branch-and-Bound and Branch-and-Cut are the most famous algorithms of this nature for solving MIP problems. These algorithms start with the linear relaxation of the problem and after solving it, they branch on one of integer variables that is taking a non-integer value at the optimal solution of the linear relaxation problem. At each node of the tree this process continues until either the optimal solution is found

* Corresponding Author. Department of Industrial and Systems Engineering, University of Wisconsin-Madison. Email: namazifar@wisc.edu

** Business Analystics and Mathematical Sciences, IBM T. J. Watson Research Center. Email: robinlh@us.ibm.com

*** Business Analystics and Mathematical Sciences, IBM T. J. Watson Research Center. Email: jjforre@us.ibm.com

or the problem is proven infeasible. This tree search is computationally expensive and there are many works that try to reduce the size of this tree resulting in solving the problem faster. Finding integer feasible solutions which are close to the optimal solution can help reduce the size of the tree significantly. Having a good feasible helps the branch-and-bound and branch-and-cut algorithms to fathom the nodes with linear programming optimal solutions which are worse than the objective value of the integer feasible solution. To find integer feasible solutions for MIP's branch-and-bound algorithm benefits from Primal Heuristic algorithms. These algorithms are designed to find integer feasible solutions for the MIP problem quickly. Many of such algorithms are in the literature [5, 6, 13, 4, 2, 11, 8, 14, 3, 1] and some of them are shown to be very effective and are being used by the MIP solvers very frequently.

Other than reducing the size of the search tree, primal heuristics are also useful when finding a reasonably good feasible solution to the MIP problem is enough and there is no need to solve the problem to optimality. For many applications locating a feasible solution which is sufficiently close to the optimal solution (based on the duality gap) is the goal, and therefore, a good primal heuristic can be very helpful.

Moreover, primal heuristics can help the bound tightening techniques by providing a good integer feasible solution. In general primal heuristics are vastly used in the comercial and free MIP solvers and there is still much effort going on to develope more effective primal heuristics.

The rest of this paper is organized as follows: in section 2 we briefly review some of the most popular primal heuristics, in section 3 we present our new primal heuristic called Pivot-and-Fix, in section 4 we talk about implementation issues and computational results of Pivot-and-Fix, and finally, section 5 cocludes the paper.

2 Primal Heuristics

There are many primal heuristic algorithms presented in the literature and each of them looks at the problem of finding integer feasible solutions for a MIP differently. To find an integer feasible solution for a MIP problem there are two general approaches [12]. The first approach tries to build up a feasible solution from scrach. Primal heuristics which use this approach are called *construction heuristics*. On the other hand, the second approach relies on the existing integer feasible solutions to find a better integer feasible solution. Primal heuristics which use this approach are called *improvement heuristics*. Here look at some of the heuristics of either of these two categories.

2.1 Construction Heuristics

We describe 2 primal heuristics and one class of primal heuristics which are based on building integer feasible solutions from scrach.

LP-and-Fix The idea behind this heuristic is very simple. Consider the original MIP problem and solve its LP relaxation. If an integer variables is taking an integer value in the optimal solution of the LP relaxation, fix its value to that integer value. After these fixing we end up with a MIP problem that is the original problem except some of its integer variables are fixed. This MIP is smaller and easier to solve than the original MIP problem and if a solution is feasible for this reduced MIP, it is also feasible for the original MIP. This is the general formulation of reduced problem:

$$
\begin{aligned}
(LP-and-Fix) \qquad & \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^p} cx+fy \\
& \text{s.t. } Ax+Gy \geq b \\
& \quad x \geq 0, y \in \mathbb{Z}^p_+ \\
& \quad y_j = \hat{y}_j \text{ for all } j \text{ with } \hat{y}_j \in \mathbb{Z}^p_+ ,
\end{aligned}
$$

in which $(\hat{x}, \hat{y})$ is the optimal solution of the LP relaxation of the problem. If feasible, solving this MIP gives us integer feasible solution(s) for the original MIP problem.

Feasibility Pump This primal heursitic was originally developed by Fischetti, Glover, and Lodi [5]. The idea here is to find an integer feasible solution by going back and forth between a feasible solution to the LP relaxation of the problem $(\hat{p})$ and an integer point $\bar{p}$ which is the closest integer point to $\hat{p}$. The algorithm first takes a point $(\hat{p})$ in the polyhedron of the LP relaxation of the problem and rounds it to an integer point (which is not necessarily feasible) $\bar{p}$. Then the algorithm finds the closest point of the polyhedron to $\bar{p}$ by solving the following problem:

$$
\begin{aligned}
(FeasibilityPump) \qquad & \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^p} \delta(\bar{p},(x,y)) \\
& \text{s.t. } Ax+Gy \geq b \\
& \quad x \geq 0, y \in \mathbb{Z}^p_+,
\end{aligned}
$$

where δ is a distance function. If the optimal value of the above problem is 0, then $\bar{p}$ is an integer feasible solution. Otherwise we replace the solution to the above problem for $\hat{p}$ and repeat the procedure.

Feasibility pump is one of the most successful primal heuristics and different varions of it is implemented and used in MIP solvers.

Searching the Relative Interior of the Polyhedron Recently there has been a few works on finding integer feasible solutions for MIP's by searching the relative interior of the polyhedron of the LP relaxation of the problem. In [ref RR] a primal heuristic called Randomized Rounding is presented which tries to find integer feasible solutions by randomly walking in the relative interior of the polyhedron of the LP relaxation of the porblem. Randomized Rounding samples from the exterem points of the polyhedron of the LP relaxation of the problem and finds random points in the convex hull of these extreme points. Then it

rounds the random points to integer points and checks their feasibility. In two other papers [10] and [ref sanjay] two different primal heuristics are presented which try to find feasible solutions by using the analytic center of the polyhedron of the LP relaxation of the problem.

2.2 Improvement Heuristics

As mentioned before, improvement primal heuristics search for better (closer to optimal) integer feasible solutions using the existing feasible solutions. Of course the drawback with these heuristics is that they can be used only if there exist incumbent solutions and they can not be used at the start of the solution of the MIP problem when we don not have any integer feasible solution. RINS [**?**], Local Branching **??**, an evolutionary method [13], and Exchange (describen in [12]). Among these heuristics here we describe RINS and Local Branching.

Relaxation Induced Neighborhood Search (RINS) RINS is a highly useful primal heuristic and might be interpreted as improvement analog of LP-and-Fix. Here suppose that $(\bar{x}, \bar{y})$ is an incumbet solution of the MIP problem. RINS solves the LP relaxation of the problem, and let's say its optimal solution is $(\hat{x}, \hat{y})$. Then RINS looks at the two points $(\bar{x}, \bar{y})$ and $(\hat{x}, \hat{y})$. If for some integer variables y_j the two values $\bar{y}_j$ and $\hat{y}_j$ are equal, then RINS fixes y_j to $\bar{y}_j$. The resulting problem is:

$$
\begin{aligned}
(RINS) \qquad & \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^p} cx + fy \\
& \text{s.t. } Ax + Gy \geq b \\
& \quad x \geq 0, y \in \mathbb{Z}^p_+ \\
& \quad y_j = \bar{y}_j \text{ for all } j \text{ with } \bar{y}_j = \hat{y}_j,
\end{aligned}
$$

which is a smaller and easier MIP compared to the original problem and if a point is an integer feasible point for this reduced MIP, it is also an integer feasible point for the original MIP problem. This technique, in fact, explores the neighborhood around the existing integer feasible solutions for beter (in terms of objective value) integer feasible solutions.

Local Branching Local branching, like RINS, explores the neighborhood of existing integer feasible solutions to find improved integer feasible solutions. The difference with RINS is instead of fixing integer variables which take the same value at the LP relaxation and the incumbent solution, local branching searches among the points that their y vector do not differ from the incumbent solution in more than k coordinates. Therefore, the local branching problem is defined as follows:

$$
\begin{aligned}
(\mathit{LocalBranching}) \qquad & \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^p} cx + fy \\
& \text{s.t. } Ax + Gy \geq b \\
& x \geq 0, y \in \{0,1\}^p \\
& \sum_{j:\bar{y}_j=0} y_j + \sum_{j:\bar{y}_j=1} (1 - y_j) \leq k \ .
\end{aligned}
$$

Here we assumed that the integer variables can only take binary values. For the general integer case the overall idea is the same. Again, we can see that this MIP is smaller than the original problem and its integer feasible solutions will be integer feasible to the original MIP problem.

3 Pivot-and-Fix

Here, in this section, we present our new primal heuristic called Pivot-and-Fix. Pivot-and-Fix tries to "explore" the extreme points of the polyhedron of the LP relaxation of the MIP in hand and look for the ones which seem "promising". By exploring a corner point we mean that we fix the value of the integer variables that are taking an integer value at that corner point to that value, and for the rest of integer variables we do a tree search. By promising we mean that for each extreme point, we look at the number of integer variables that are taking integer values. This number can represent some facts about the size of the neighborhood of that extreme point that needs to be explored. If this number is too big, it would suggest that this extreme point might not be worth exploring, simply because the neighborhood of the extreme point that need to be explored is too big. On the other hand if this number (number of integer variables that are taking integer values) is too small, then the neighborhood of the extreme point that needs to be searched is too small, and most likely does not include any integer feasible solution.

To find the extreme points of the polyhedron of the LP relaxation of the MIP, we generate random objective functions and set them as the objective function of the LP relaxation of the problem. We also determine the sense of the objective function based on the sense of the constraint we tilted to get the objective function (maximization for $\leq$ and minimization for $\geq$ and $=$). Then Pivot-and-Fix starts solving this linear program using primal simplex. At each iteration of primal simplex we get an extreme point of the polyhedron of the LP relaxation. The way we generate the random objective funstions is we take one of the constraints of the problem and we randomly tilt this constraints by randomly adding (or subtracting) a fraction of each coeffint of the constraint to (or from) itself. The reason behind tilting the constraints is that firstly, we want to make the problem to have a single optimal solution, and secondly, we want to avoid the cases in which several of these randomly generated problems have the same optimal solutios. If such cases happen, we will not have enough extreme points to explore.

As we mentioned above, to find extreme points of the polyhedron of the problem we randomly generate linear objective functions and solve these LP's using primal simplex. At each iteration of primal simplex, Pivot-and-Fix looks at the number of integer variables that are NOT taking integer values at that extreme point. If this number is less than a threshold, it means that we have enough integer variables to fix (we will talk about this threshold in more details later). Then we start fixing those integer variable that are taking an integer value to that value untill the number unfixed integer variables is less than the threshold. This new problem is the same as the original MIP problem except that some of its integer variables are fixed and hence is smaller than the original problem. For this problem we start a branch-and-bound search with a limited number of nodes (because we do not want to spend too much time on one subproblem). If branch-and-bound finds a feasible solution, this is a feasible solution for the original problem. As a result we add a new constraint with the left hand side the same as the original objective function, the sense $\leq$, and the right hand side the objective value of the newly found feasible solution. By doing this we make Pivot-and-Fix to look for better feasible solution in the next steps.

Here is the overall view of the Pivot-and-Fix algorithm:

– REPEAT

- Generate a random objective function
- Using primal simplex, solve an LP with the original problem's constraints and the randomly generated objective function
- At each simplex iteration, check the number of integer variables that are not taking an integer value
- If this number is less than a threshold, fix those variables and do a branch-and-bound for the rest.
- If the branch-and-bound finds a feasible solution:
  - Save the solution as an integer feasible solution for the original MIP
  - Add a new $\leq$ constraint to the problem with the original objective coefficient and the objective value of the feasible solution on the right hand side

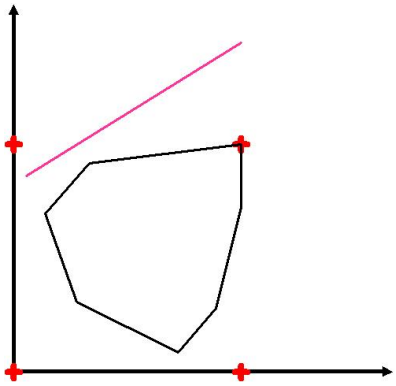

Fig. 1: The feasible region of the problem and the original objective function. Both of the variables are binary and there is only one integer feasible solution which is the optimal solution of the problem.

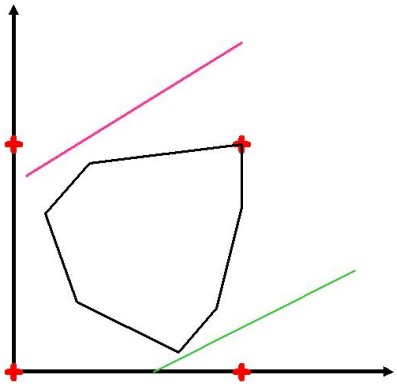

Fig. 2: Generate a random objective function by tilting one of the constraints. Set up a linear program with the randomly generated objective function and the constraints of the original problem.

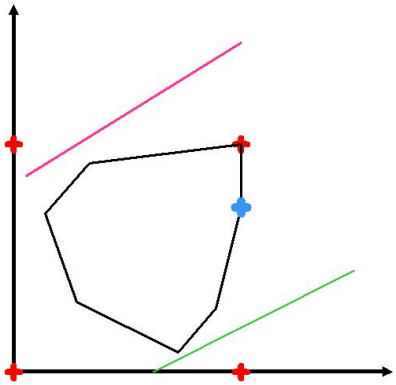

Fig. 3: Solve the LP using primal simplex. At each simplex iteration, check the number of integer variables that are not taking integer values. If this number is bigger than the threshold go to the next simplex iteration; else fix the integer variables that are taking integer values.

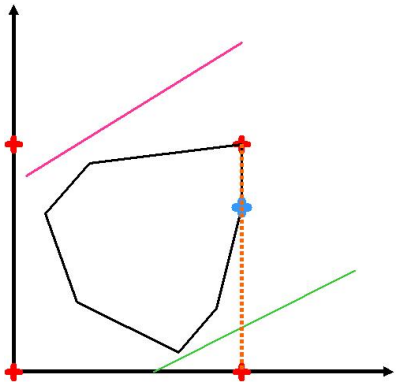

Fig. 4: Using branch-and-bound start solving the MIP subproblem which we get by fixing some of the integer variables to their integer values at the extreme point. Do the branch-and-bound until a certain number of tree nodes reached.

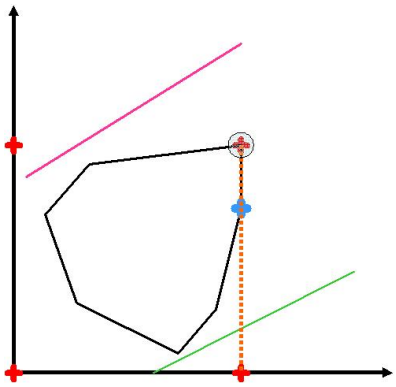

Fig. 5: If a feasible solution is found in the course of branch-and-bound, this solution is also feasible for the original problem.

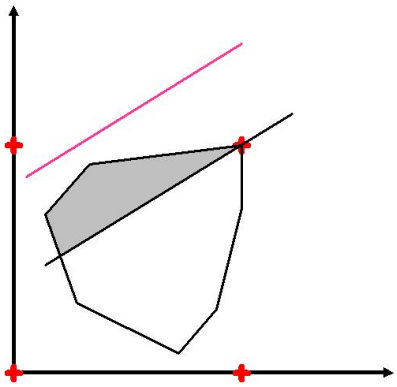

Fig. 6: Add the constraint which cuts off any point with worse objective value than the objective value of the feasible solution just found.

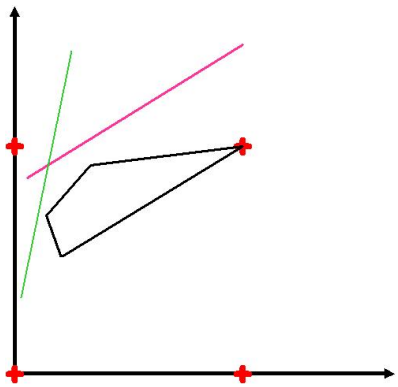

Fig. 7: Generate another random objective function and repeat.

Here we would like to address some issues that need to be taken into consideration in order for Pivot-and-Fix to perform satisfactorily. First of all, as we mentioned earlier, we randomly generate objective functions by tilting the constraints of the MIP. One point to have in mind here is usually in MIP models consecutive constraints are somehow related to each other in the sense that they might have a nonempty intersection in the polyhedron of the problem. This fact might lead the two linear programs with objective functions which are built by tilting two consecutive constraints have the same (or very similar) objective values. If we solve the linear program with the objective funstion made by one the

consecutive constraints right after solving the linear program with the objective function made by the other constraint, most likely we will not get many different extreme point, considering the warm start of the LP solves. Therefore, we pick the constraints randomly, and then tilt them to build an objective function.

Another point to remember here is there is no need to solve the linear programs we generate all the way to optimality. If after a large number of primal simplex iteration we still have not found any extreme point with the right number of integer variables that are taking integer values, we could simply quit solving the current linear program and go to the next one.

Moreover, as we mentioned before, we need to discuss about the threshold we introduced earlier for the number of integer variables that are not taking integer values at an extreme point. This threshold was introduced to have a touchstone to evaluate how likely extreme point is close to an integer feasible point. This parameter, perhaps, is the trickest parameter in Pivot-and-Fix. One very important factor in setting this threshold is the number of integer variables in the problem. Experiments show that the fewer the number of integer variables, the higher this threshold should be. For instance for MIP problems with more than 1000 integer variables, maybe 10 percent of the number of integer variables is a good threshold. But 10 percent is not a good threshold for a problem with 100 variables, because it leaves us with only 10 unfixed integer variables; whereas in this case we would probably would like to have a lot more unfixed integer variables. The other possibility is dynamically changing the threshold based on our observations during the run of Pivot-and-Fix. As we mentioned before, we set a number as the maximum number of nodes that Pivot-and-Fix explores at each branch-and-bound for each subproblem. No imagine that for several consecutive subproblems, the branch-and-bound hits that maximum node number and it does not find any feasible solution. It means that maybe our threshold is too high and we might want to reduce it. Also, in another case, imagine that for several consecutive subproblems the branch-and-bound finishes before reaching the maximum number of nodes and it finds one or more feasible solutions. This might suggest us that we could increase the treshold to provide Pivot-and-Fixmore opportunity to find feasible solutions.

Also we might want to dynamically change the maximum number of branch-and-bound nodes Pivot-and-Fix explores. If for several consecutive subproblems the branch-and-bound finds a feasible solution but it also reaches the maximum number of nodes, we might want to increase the maximum number of nodes we explore at each branch-and-bound.

4 Implementation and Computational Results

4.1 Implementation and Test Bed

We implemented Pivot-and-Fix using COIN-OR Branch-and-Cut (Cbc) [7] which is an open source MIP solver. In this implementation we came up with a dynamic system to manage the threshold of the number of integer variables that are not

taking integer values, and maximum number of nodes to explore in branch-and-bound while solving the subproblems. To test the performance of Pivot-and-Fix we chose MIPLIB 2003 [9] library of MIP problems which is the most frequently used set of MIP problems in the MIP literature. MIPLIB 2003 contains a diverse set of problems with different difficulty levels (hard, medium, easy). The MIPLIB 2003 library consists of 60 problems, 5 of which have not been solved to optimality, 19 of which have been solved (in more than an hour), and 36 of which have been solved whithin an hour.

For some of the MIPLIB 2003 instances solving the LP relaxation of the problem takes a long time and this can affect the performance of Pivot-and-Fix on those instances because Pivot-and-Fix solves many linear programs. This suggests that Pivot-and-Fix is not suitable for the problems whith large LP relaxations. It is not too hard check wether an LP instance is hard to solve using the information like number of constraints, number of variables, number of nonzero elements in the matrix of the probm.

To test the performance of Pivot-and-Fix , from MIPLIB 2003 we picked the MIP instances for which Cbc takes more than a minute to solve. Because such problems are already being solved very quickly and are too easy for the current solvers to solve. Moreover 11 of instances have very large LP relaxations which are not good candidates for Pivot-and-Fix. For the rest of the instances we compare the quality of the feasible solution and the time it takes to get the feasibles solutions of our heuristic against the default Cbc-2.2. The experiments are done on .....[reka].... In the default Cbc 6 primal heuristics are on which are: Feasibility Pump, RINS, Diving, Combine, Rounding, and Greedy. Table 1. shows the results of these runs. In this table the objective value of the solution and the time taken to find that solution for Cbc and Pivot-and-Fix are presented. The last column of the table shows the optimal value of the instance. As you can see for many of the instances Pivot-and-Fix finds a better solution faster compared to Cbc. For some of the instances Cbc finds a better solution in a shorter time and for some of the instances one finds a better solution while the other finds a solution faster. Here are a few comments about the runs:

- danoint: After 10 minutes Cbc's best solution is 68.75.
- liu: After 6 minutes Cbc's best solution is 4490.
- mas74: After 8 seconds Cbc'c best solution is 13915.9.
- mod011: After 72 seconds Cbc'c best solution is -4.70938e+07.
- modglob: After 11 seconds Cbc'c best solution is 20784300.
- noswot: After 170 seconds Cbc's best solution is -40.
- nw04: After 51 seconds Cbc's best solution is 18016.
- pk1: After 21 seconds Cbc's best solution is 21.
- qiu: After 55 seconds Cbc's best solution is -119.65.
- markshare2: After 17 seconds Cbc's best solution is 83.
- glass4: No heuristic finds a feasible solution.
- net12: After 200 seconds Cbc did not find anything.
- nsrand-ipx: After 240 seconds Cbc's best is 67680.
- tr1230: After 188 seconds Cbc's best is 233784.

– protfold: After 5 minutes Cbc did not find any feasible solution.

Table 1: Comparing Randomized Rounding against default Cbc for mixed integer knapsack problems

	COIN-Cbc 2.2		Pivot-and-Fix		
Problems	Solution	Time	Solution	Time	Optimal Solution
aflow30a	1974	42.53	**1393**	**2.23**	1158
aflow40b	1974	41.09	**1548**	**6.13**	1168
air04	**59624**	**2.46**	-	-	56137
air05	**27237**	**2.44**	-	-	26374
arki001	75956600	41.59	**7596533.28**	**4.12**	7.58E+6
cap6000	**-2.45E+6**	**1.29**	-2214561	2.43	-2.45E+6
danoint	85	27.87	**66.5**	**10.52**	65.67
fast0507	212	**26**	212	44	174
gesa2	25891400	15.27	**25803948**	**1.38**	25779900
harp2	**-71338900**	27.05	-63156324	**0.24**	-73899800
liu	4490	**2.91**	**4348**	4.54	?
mas74	14372.9	0.13	**13725.48**	0.17	11801.2
mas76	40560.05	0.9	40560.05	0.9	40005.1
mkc	-281.25	0.76	**-389.65**	0.53	-563.85
mod011	-4.28E+7	**0.41**	**-53656254**	7.32	-54558500
modglob	20784600	**0.1**	**20762314**	1.5	20740500
noswot	-40	0.15	**-41**	0.15	-41
nw04	19124	**2**	**17172**	4.68	16862
pk1	23	0.13	**18**	0.64	11
qiu	-10.03	38.67	**-128.47**	**6.62**	-132.87
rout	**1217.18**	0.26	2375.25	**0.05**	1077.56
timtab1	1.29E+6	20.24	**958589**	**3.93**	764772
timtab2	1.71E+6	80.23	**1600627**	**28.81**	?
markshare1	77	15.16	**73**	**0.09**	1
markshare2	83	0.81	**78**	**0.09**	1
gesa2-o	2.61E+7	26.1	**25903660.96**	**5.96**	2.58E+7
glass4	**2.33E+9**	9.23	2833356866.66	**0.3**	1.20E+9
misc07	**3160**	9.42	2810	**3.22**	2810
net12	-	-	**337**	**17.2**	214
nsrand-ipx	67680	**2.42**	**66080**	28.12	51200
protfold	-	-	-	-	-31
seymour	**434**	8.41	451	**0.37**	423
set1ch	**76592.5**	0.15	95285	0.19	54537.8
tr1230	**264195**	**0.54**	215705	8.2	130596
sp97ar	1.20E+9	17.14	**1008600797.26**	**3.06**	?
roll3000	**14813**	94.64	15718	**9.7**	467.41

As Table 1. and the comments on some of the instances suggest, it seems that Pivot-and-Fix can find reasonably good feasible solution quickly and in general

seems very likely that it could help to improve the performace of mixed integer programming solvers.

5 Conclusions

In this paper we presented Pivot-and-Fix , a new primal heuristic for mixed integer programming. This primal heuristics tries to find extreme points of the polyhedron of the LP relaxation of the MIP at which the number of integer variables that are not taking integer values is less than a threshold. For such extreme points we fix the integer variables that are taking integer values and for the rest of the integer variables we start a branch-and-bound process. Any feasible solution that is found by these brnach-and-bound processes is feasible to the original MIP, as well. We implemented this heuristic using COIN-OR and compared it with COIN-Cbc. The results suggest that Pivot-and-Fix is able to improve the performance of MIP solvers in general, and COIN-Cbc specifically, significantly.

Moreover, there might be a possibility of using the extreme points of the polyhedron that Pivot-and-Fixvisits to generate mixed integer Gomory cuts, which need to be investigated.

References

[1] T. Achterberg and T. Berthold. Improving the Feasibility Pump. *Discrete Optimization*, 4(1):77–86, 2007.
[2] E. Balas and C. Martin. Pivot and complement - a heuristic for 0-1 programming. *Management Science*, 26:86–96, 1980.
[3] E. Balas, S. Schmieta, and C. Wallace. Pivot and shifta mixed integer programming heuristic. *Discrete Optimization*, 1(1):3–12, 2004.
[4] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1(A)):71–90, 2005.
[5] M. Fischetti, Glover F., and A Lodi. The Feasibility Pump. *Mathematica Programming: Series A and B*, 104(1):91–104, 2005.
[6] M. Fischetti and A. Lodi. Local Branching. *Mathematical Programming*, 98:23–47, 2003.
[7] J. Forrest. CBC, 2004. Available from `http://www.coin-or.org/`.
[8] F. Hillier. Efficient heuristic procedures for integer linear programming with an interior. *Operations Research*, 17(4):600–637, 1969.
[9] A. Martin, T. Achterberg, and T. Koch. MIPLIB 2003, 2003. Avaiable from `http://miplib.zib.de`.
[10] J. Naoum-Sawaya and S. Elhedhli. Finding Feasible Solutions in Mixed Integer Programming Using an Interior Point Cutting Plane Method. *Technical Report, Department of Management Sciences, University of Waterloo*, 2009.
[11] M. Nediak and J. Eckstein. Pivot, cut, and dive: A heuristic for 0-1 mixed integer programming. *Jounal of Heuristics*, 13(5):471–503, 2007.
[12] Y. Pochet and L. Wolsey. *Production Planning by Mixed Integer Programming*. Springer, New York, NY, USA, 2006.

[13] E. Rothberg. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. *INFORM Jounal of Computing*, 19(4):534–541, 2007.
[14] M. Saltzman and F. Hillier. A Heuristic Ceiling Point Algorithm for General Integer Linear Programming. *Management Science*, 38(2):26–283, 1992.