An Empirical Study of the Stable Marriage Problem with Ties and Incomplete Lists [1]

Ian Philip Gent [2] and **Patrick Prosser** [3]

Abstract. We present the first complete algorithm for the SMTI problem, the stable marriage problem with ties and incomplete lists. We do this in the form of a constraint programming encoding of the problem. With this we are able to carry out the first empirical study of the complete solution of SMTI instances. In the stable marriage problem (SM) [**?**] we have n men and n women. Each man ranks the n women, giving himself a preference list. Similarly each woman ranks the men, giving herself a preference list. The problem is then to marry men and women such that they are *stable* i.e. such that there is no incentive for individuals to divorce and elope. This problem is polynomial time solvable. However, when preference lists contain ties and are incomplete (SMTI) the problem of determining if there is a stable matching of size n is then NP-complete, as is the optimisation problem of finding the largest or smallest stable matching [**?**, **?**]. In this paper we present constraint programming solutions for the SMTI decision and optimisation problems, a problem generator for random instances of SMTI, and an empirical study of this problem.

1 Introduction

In the stable marriage problem [**?**] we have n men and n women. Each man ranks the n women, giving himself a preference list. Similarly each woman ranks the men, giving herself a preference list. The problem is then to marry men and women such that they are *stable*. By stable we mean that there is no incentive for individuals to divorce and elope. For example, a matching would be unstable if it contained the marriages Romeo to Isobel and John to Juliet, where Romeo prefers Juliet to Isobel, and Juliet prefers Romeo to John, i.e. Romeo and Juliet would elope. This problem has a long history, and an optimal algorithm was proposed by Gale and Shapley almost 40 years ago [**?**]. The algorithm's complexity is $O(n^2)$, and is linear in the size of the problem, where size is measured in terms of the n people each with a preference list of size n.

If men or women find some members of the opposite sex unacceptable, preference lists become incomplete. These problems are classified as stable marriage problems with incomplete lists (SMI) and are again solvable in polynomial time. We might also have ties in the preference lists. That is, a man (or a woman) might be indifferent between a number of his (or her) choices. For example John might have a preference such that he prefers Isobel to Jane, but Jane ties with Susie. In the extreme when all potential partners tie with one another, we are asking only for a matching and stability is not an issue. However, when we combine ties with incompleteness, we get the stable marriage problem with ties and incomplete lists (SMTI). We restrict ourselves in this paper to weak stability: under this definition a marriage is unstable if there is a man m_i and a woman w_j, each of whom *strictly* prefers the other to his/her current partner [**?**], i.e. m_i and w_j will elope.

No complete algorithm has previously been proposed for SMTI and no empirical study has been carried out. The question "Is there a weakly stable matching?" is uninteresting as there always is [**?**]. So in this paper we present a constraint programming encoding for the SMTI decision problem "Is there a stable matching of size n?" and the optimisation problem, to find the largest or smallest stable matching. These questions are NP-complete [**?**, **?**]. We also propose a problem generator for random instances of SMTI. We then study the SMTI investigating what features of the problem appear to influence the hardness and size of stable matchings.

The paper is organised as follows. In the next section we present a problem generator for SMTI. We then present constraint programming solutions for the SMTI decision problem and optimisation problem. These are the first complete algorithms for these problems. We then discuss the constrainedness of SMTI. The empirical study is then presented and the paper concludes.

2 Random Instance Generation

A class of randomly generated instance of SMTI is represented by a triple $\langle n, p_1, p_2\rangle$ where n is the number of men and women in the problem, p_1 is the probability of incompleteness and p_2 is the probability of ties. Problems are generated as follows

1. A random preference list of size n is produced for each man and each woman.
2. We iterate over each man's preference list as follows. For a man m_i and for all women w_j in his preference list, we generate a random number $0 \leq p < 1$. If $p \leq p_1$ we delete w_j from m_i's preference list and delete m_i from w_j's preference list.
3. If any man or woman has an empty preference list, discard the problem and go to step 1.
4. We iterate over each person's (men and women's) preference list as follows. For a man m_i and for his choices c_i ranging from his second to his last, we generate a random number $0 \leq p < 1$. If $p \leq p_2$ then the preference for his c_i^{th} choice is the same as his c_{i-1}^{th} choice, otherwise his c_i^{th} choice is one greater than his c_{i-1}^{th} choice.

An instance generated as $\langle n, 0.0, 0.0\rangle$ will be a stable marriage problem with a stable matching of size n. An instance $\langle n, 0.0, 1.0\rangle$ is a stable matching problem with complete indifference, and there are $n!$ matchings. An instance $\langle n, 1.0, p_2\rangle$ is a problem with empty preference lists, and is obviously unsolvable. Figure **??** shows a randomly

[1] This work was supported by EPSRC research grant GR/M90641.
[2] School of Computer Science, University of St. Andrews, North Haugh, St. Andrews KY16 9SS, Scotland, email:ipg@dcs.st-and.ac.uk
[3] Computing Science, Glasgow University,17 Lilybank Gardens, Glasgow G12 8RZ, Scotland, email:pat@dcs.gla.ac.uk

Men's lists	Women's lists
1: 2 (6 4)	1: (5 3) 6
2: (2 5) 6	2: 2 5 1 6
3: 1 3 6	3: (3 4)
4: 6 3	4: 6 1
5: 2 1 5	5: 5 2 6
6: 6 (4 2) 5 1	6: 1 (4 6) 2 3

Figure 1. A randomly generated SMTI instance with 6 men and 6 women, generated with parameters $\langle 6, 0.5, 0.25\rangle$. The instance has a largest stable matching of size 6 namely (4,2,1,3,5,6), and a smallest stable matching of size 5, namely (4,2,-,3,1,6)

generated SMTI $\langle 6, 0.5, 0.25\rangle$. Preferences that tie are in braces. The largest stable matching is of size 6 and the smallest is of size 5.

3 A Constraint Programming Representation

Two representations are presented, the first being for the decision problem *Is there a stable matching of size n?*, and the second for the optimisation problem *What is the size of the largest (smallest) stable matching?*.

The encoding is a simple extension of the SMI encoding presented in Section 2 of [**?**] and uses $2n$ variables, where the domain of a variable corresponds to a preference list. For the i^{th} man (woman) we have the variable m_i (w_i). If there is a value $j \in domain(m_i)$ then woman w_j is in the man's preference list. The man has a preference $pref(m_i, j)$ for woman w_j, where $1 \leq pref(m_i, j) \leq n$. A stable marriage constraint exists between man m_i and woman w_j when $j \in domain(m_i)$ and $i \in domain(w_j)$. The stable marriage constraint is represented extensionally as a set S of infeasible pairs (nogoods). A nogood (a, b) is in S if it corresponds to a blocking pair or a relationship which is not a marriage. That is, (a, b) is a blocking pair if m_i prefers w_j to w_a and w_j prefers m_i to m_a i.e. $a \in domain(m_i) \wedge b \in domain(w_j) \wedge pref(m_i, a) < pref(m_i, j) \wedge pref(w_j, b) < pref(w_j, i)$. (Note that the strict inequalities correspond to the definition of weak stability.) The pair $(a, b) \in S$ is not a marriage if m_i marries woman w_j but woman w_j marries someone else, or woman w_j marries man m_i but m_i marries someone else, i.e. $a \in domain(m_i) \wedge b \in domain(w_j) \wedge (a = j \wedge b \neq i \ \vee \ a \neq j \wedge b = i)$.

	1	4	6	2	3
6	x	x		x	x
4			x	x	x
2			x	x	x
5			x	x	x
1			x	x	x

Figure 2. The conflict matrix for constraint $C_{6,6}$ from the problem in Figure **??**

Figure **??** shows the conflict matrix corresponding to the stable marriage constraint acting between man m_6 and woman w_6 in the problem of Figure **??**. The domain values of the variables have been listed in preference order, exposing the structure first identified in [**?**].

When preference lists are incomplete it might not be possible to find a matching of size n. Therefore, it might be worthwhile adding two redundant $allDiff$ constraints [**?**] to detect when there are insufficient values to allow all men and all women to be married. In the empirical study we will investigate the behaviour of this redundant constraint.

When no complete stable matching can be found we might then be interested in finding the largest or the smallest stable matching, i.e we have an optimisation problem. The above encoding is then modified as follows. We add a virtual person $n + 1$ to every man and woman's domain with a preference value of $n + 1$, i.e. $pref(m_i, n + 1) = n + 1$. Consequently, a person prefers to be married than to be single. In addition we associate a zero/one variable to each man and each women. The zero/one variable takes a value of one if the person is married, otherwise it is zero. That is $mm_i = 0 \leftrightarrow m_i = n + 1$ and $wm_j = 0 \leftrightarrow w_j = n + 1$, where mm_i and wm_i are in $\{0, 1\}$, and $mm_i = 1$ ($wm_j = 1$) can be read as *man m_i (woman w_j) is married*, and $\leftrightarrow$ is the biconditional. To find a largest stable matching we maximise the sum of the mm variables, and to find the smallest stable matching we minimise the sum.

There are n variables each with domain size n. There is a binary constraint from each man to all women, i.e. n^2 binary constraints, and each of these constraints contains $O(n^2)$ nogood pairs. Therefore the size of the encodings is $O(n^4)$. The complexity of arc consistency is $O(e.d^r)$ where e is the number of constraints, d is the domain size, and r is the arity of the constraints [**?**, **?**]. Consequently the cost of enforcing arc consistency in our encoding is $O(n^4)$. The $O(n^2)$ encoding proposed in [**?**] has not yet been extended to handle ties.

For the above encodings the search process uses a variable and value ordering heuristic. Only person variables are selected (i.e. the zero/one variables are not selected for instantiation), preference being given to the variable with the least remaining values in its domain i.e. the *minimum remaining values* variable ordering heuristic. Values are then selected in preference order, such that a person attempts to marry his or her most preferred partner. This value ordering heuristic guarantees a failure free enumeration of solutions in SM [**?**].

When presented with an instance of SMI, i.e. problems generated as $\langle n, p_1, 0.0\rangle$, the above encoding for the decision problem reverts to polynomial performance, and with the value ordering heuristic we are guaranteed failure free enumeration of all stable matchings. This is because the encoding reduces to the SMI encoding of [**?**] for which search is polynomial.

4 The Constrainedness of SMTI

In [**?**] a measure of constrainedness was proposed. The measure of constrainedness (kappa) is defined as $\kappa = 1 - \frac{log(\langle Sol\rangle)}{log(|S|)}$, where $\langle Sol\rangle$ is the expected number of solutions, and $|S|$ is the size of the state space. κ is defined for an ensemble of problems. When every state is a solution $\langle Sol\rangle = |S|$ and $\kappa = 0$, and problems are easy and soluble. When no state is a solution $log(\langle Sol\rangle) = -\infty$ and $\kappa = \infty$, and problems are again easy but unsolvable. When on average each problem has one solution $log(\langle Sol\rangle) = 0$ and $\kappa = 1$, problems will be on the knife edge of constrainedness, where we expect 50% to be soluble and most to be hard.

For a constraint satisfaction problem κ is defined to be $-\frac{\sum_{c \in C} log(1-p_c)}{\sum_{v \in V} log(m_v)}$, where C is the set of constraints, V is the set of variables, p_c is the tightness of a constraint, and m_v is the size of the domain of the variable v. Given a constraint C involving a set of variables, the tightness of that constraint p_c can be calculated as the number of infeasible tuples divided by the number of possible tuples for those variables.

By representing SMTI as a constraint satisfaction problem we can measure κ for each instance generated. This will give us some indication of what ensemble such an instance most likely belongs to. However, we can make some conjectures as to how the difficulty of SMTI will vary as we vary the problem generation parameters $\langle n, p_1, p_2 \rangle$. When we increase p_2 we should expect each stable marriage constraint to become looser, i.e. the number of infeasible tuples will fall. Therefore κ should be inversely related to p_2. When we increase p_1 this will increase the amount of incompleteness in preference lists. Consequently domain sizes will fall, and we should expect κ to rise. However, as domain sizes fall so too does the number of stable marriage constraints. Therefore, it is not immediately clear if this fall in the number of constraints will win out against the falling domain sizes. Will κ fall or rise with p_1? And what will happen as we vary p_1 and p_2 together? Will these be opposing forces, where p_1 tends to drive problems towards insolubility, whereas p_2 tends to make problems looser? We will investigate these questions in the next section.

5 The Empirical Study

We performed our experiments using the choco constraint programming toolkit [**?**]. The study is mostly of problems of size 10. Problems were generated with incompleteness p_1 varying from 0.1 to 0.8 in steps of 0.1. When $p_1 \geq 0.9$ problems have empty preference lists, and are rejected from this study. For each value of p_1 we vary ties p_2 from 0.0 to 1.0 in steps of 0.01, with a sample size of either 100 or 50 at each data point. Experiments were run on machines with either 733MHz or 1GHz processors, with between 256MB and 1GB of RAM. The experiments reported here took in excess of 2 months CPU time. We also coded an independent implementation, written by a different author in Eclipse, and obtained consistent results with those presented here. In our experiments we first investigate how parameters p_1 and p_2 influence the decision problem "Is there a stable matching of size n?". We then explore the optimisation problem "What is the size of the largest and the smallest stable matchings?".

5.1 The Decision Problem

In the decision problem we determine if there is a stable matching of size n. This is a feature of the problem, and is algorithm independent. Figure **??** shows for each value of p_1 the proportion of soluble instances as we vary the amount of ties p_2. We see that as the amount of ties p_2 increases the proportion of soluble instances increases. This suggests that as we increase ties the constraints between men and women become looser, consequently we should expect to see a fall in the constrainedness of problems. We also observe that as p_1 increases, i.e. preference lists get shorter, solubility decreases. This might at first appear unsurprising. However, as preference lists get shorter the number of stability constraints fall. This fall is not enough to prevent a fall in solubility due to falling domain size.

In Figure **??** we plot solubility against the average constrainedness of the problem instances, i.e. κ is on the x-axis. We see the familiar phase transition behaviour as observed in [**?**, **?**].

The allDiff constraint makes no difference. The number of search nodes was the same with and without this redundant constraint, and there was no significant difference in run times. Figure **??** shows the average cost of answering the decision problem, measured in terms of search nodes, for $\langle 10, p_1, p_2 \rangle$ plotted against p_2. Search costs increases as we increase ties p_2. This is because constraints get looser as ties increase, consequently the problem is less determined by propagation. Therefore at each instantiation a choice has to be made. Nev-

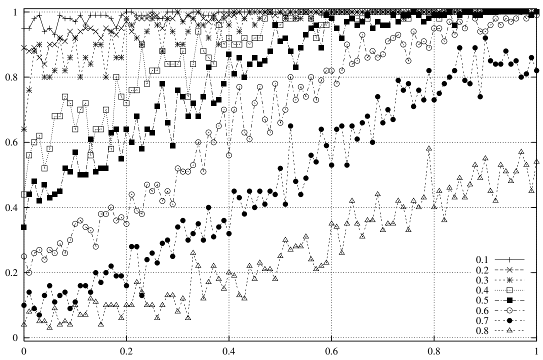

Figure 3. The decision problem: is there a stable matching of size n?

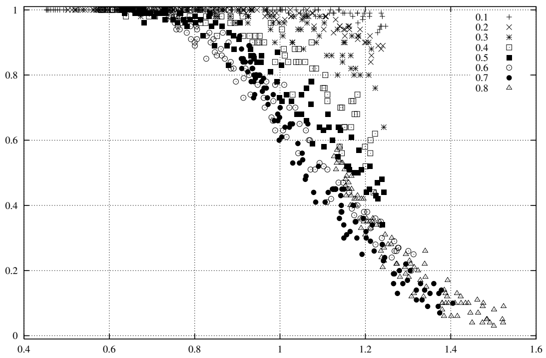

Figure 4. The decision problem: is there a stable matching of size n for a given value of κ?

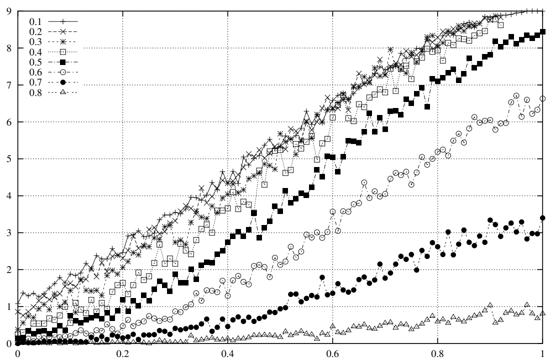

Figure 5. The average cost of the decision problem

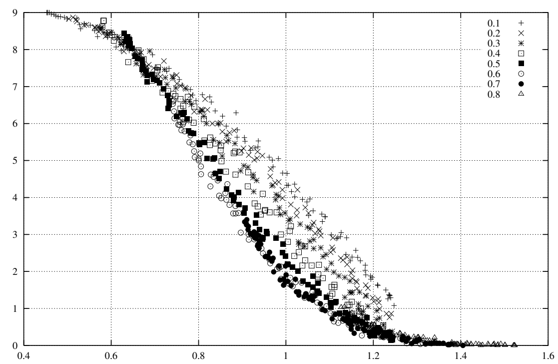

Figure 6. The average cost of decision problem, plotted against κ

ertheless, search effort is small, never more than 9 nodes. In addition we see that search cost decreases as we increase incompleteness p_1.

Figure **??** re-plots the above data, this time against the average value of κ, rather than p_2. The contour is somewhat surprising, with search effort falling with increasing constrainedness. There is no sign of a complexity peak normally associated with the solubility phase transition.

Our final figure in this section, Figure **??**, shows how search effort varies as we vary problem size. There are 6 contours for $\langle n, 0.5, p_2\rangle$, with n equal to 10 to 60 in steps of 10. The median search cost in nodes is plotted against p_2. We plot against p_2 rather than κ because we observed a systematic bias in the values of κ since the degree of a variable is proportional to np_1. It would be interesting future work to define a specialised value of κ for SMTI, not based on the constraint encoding, to avoid this problem. It appears that median search effort increases polynomially with problem size. However, we observed occasional hard problems. For example, an instance of $\langle 60, 0.5, 0.61\rangle$ took 15438 nodes.

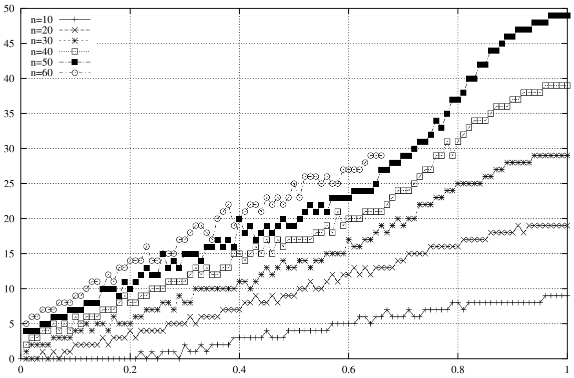

Figure 7. The median cost of the decision problem for $\langle n, 0.5, p_2\rangle$, plotted against p_2.

5.2 Optimisation

We now investigate the size of the largest and smallest stable matchings. Stability is an 'interpolating invariant' [**?**]: that is, there is stable matching of every intermediate size between the min and max. The maximum sized matching is never more than twice the size of the minimum [**?**].

Throughout this section we plot against p_2 instead of κ, since constrainedness is defined for the decision problem, not the optimisation question. Figure **??** shows the average size of the minimum and maximum stable matchings for varying p_2 and $p_1 = 0.5$. These are initially the same size, as they must be, for $p_2 = 0$, but the difference increases with p_2 until for $p_2 = 1$ there is an average difference in size of more than 3. At complete indifference this corresponds to the difference in size of maximal matchings. We observed a similar pattern at different values of p_1. At complete indifference the largest mean difference is of 3.5 at $p_1 = 0.7$, after which the shortening preference lists reduces the size of the maximum matching.

We now examine the computational cost (measured as search nodes explored) of finding the minimum and maximum stable matchings. These were computed in separate runs. Figure **??** shows the cost of finding the largest matching and verifying its maximality, for varying p_1, p_2. Notice that we have used a log scale for cost. Cost increases as we increase p_2 for given p_1. We explain this by noting that

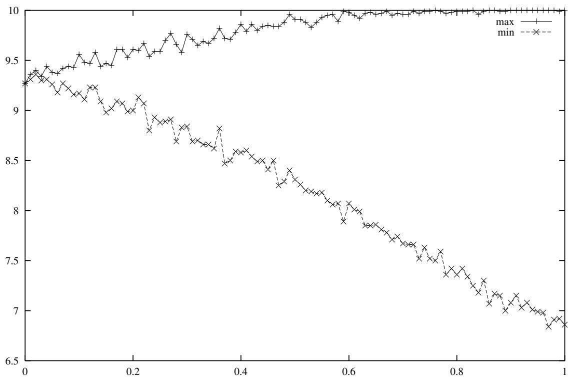

Figure 8. The average size of the smallest and largest stable matchings for $\langle 10, 0.5, p_2\rangle$

for high p_2, the large number of ties leads to many candidate matchings to explore. In the region of high p_2 we see noisy behaviour in the search cost. As yet, we have no explanation of why we are seeing these occasionaly hard problems.

Figure **??** shows the cost of finding (and proving optimal) the smallest matching. Here, search cost (again shown on a log scale) increases exponentially with p_2, with numbers of nodes in the millions for $p_1 = 0.1$, $p_2 = 1$. This is because of the factorial number of matching required to be eliminated as possible minimal matchings when verifying minimality. Remembering that when $p_2 = 1$ the problem is a simple matching problem and is in P, it is disappointing that the search has this property. Clearly, some new constraint is required to eliminate this thrashing if SMTI instances with large p_2 are found in practice.

Figures **??** and **??** show how these search costs scale as we increase n with $p_1 = 0.5$. We could not test high values of p_2 at larger n because of the large runtimes mentioned above. We see similar behaviour to that we noted at $n = 10$, with noise in the search cost for largest size, and clear exponential growth in the cost for smallest size. It is less clear, because of the noise, whether the cost to find the maximal matching is growing polynomially or exponentially. Certainly we do not see strong evidence of the median growing exponentially. It may be polynomial, as our better evidence for the decision problem showed.

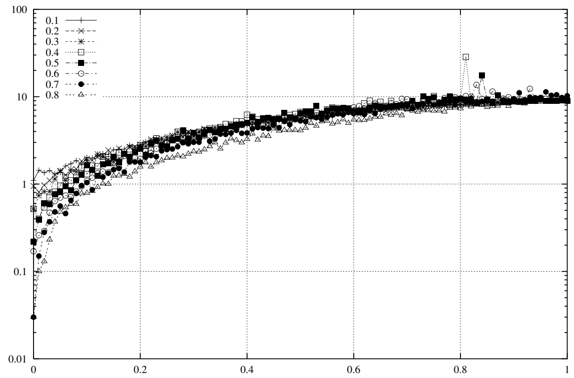

Figure 9. The average cost, in nodes, of finding the largest stable matchings for $\langle 10, p_1, p_2\rangle$

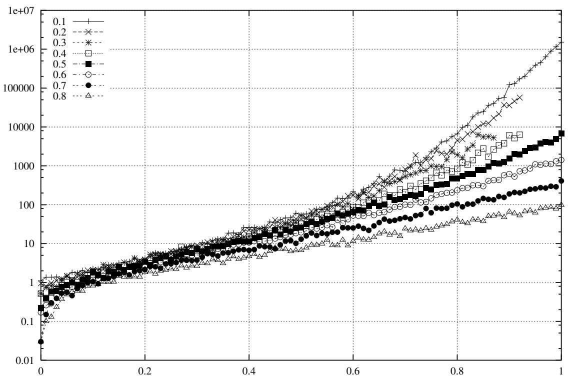

Figure 10. The average cost, in nodes, of finding the smallest stable matchings for $\langle 10, p_1, p_2 \rangle$

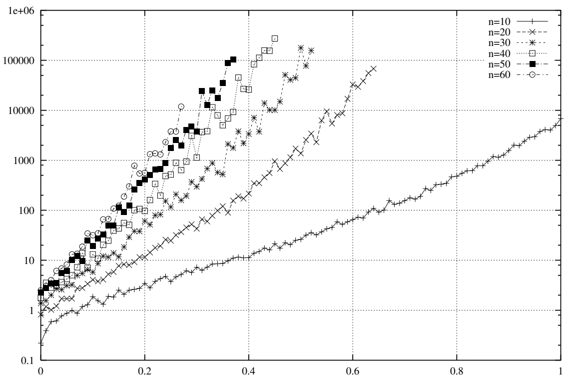

Figure 12. The average cost, in nodes, of finding the smallest stable matchings for $\langle n, 0.5, p_2 \rangle$

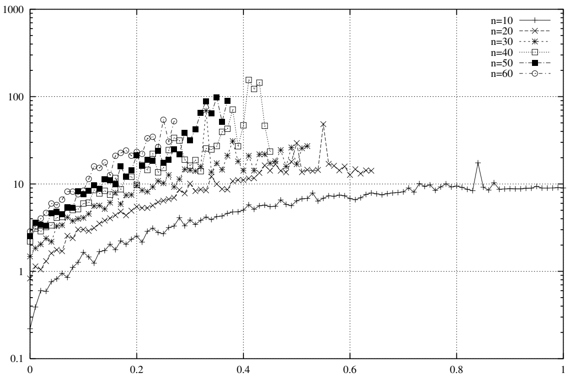

Figure 11. The average cost, in nodes, of finding the largest stable matchings for $\langle n, 0.5, p_2 \rangle$

6 Conclusions and Further Work

We have reported on a notable success for constraint programming (CP). We have been able to implement a complete algorithm for the SMTI problem and perform an empirical study using "off the shelf" CP technology. It was with this general technology that we achieved our results, rather than develop special purpose code for our study of SMTI. Our study has shown how the size of stable matchings and the decision and optimisation costs vary as we vary the random generation parameters. Our random instances show significant differences in size between smallest and largest, an important feature of the SMTI problem.

We have opened many interesting avenues for further research. First, why did we fail to observe a complexity peak at the solubility phase transition? Might this emerge when we look at bigger instances? Second, why was the cost of finding the smallest stable matching considerably harder than finding the largest stable matching? Might a better encoding of SMTI result in a reduction in cost for the minimisation problem? Thirdly, determining if there is a stable matching of size n is polynomial time solvable for SM, SMI, and SMT, yet it is NP-Complete for SMTI. Consequently, there must be a phase transition from P to NP-completness when both p_1 and p_2 are greater than zero. When will we start to see problems behave as if they are NP-Complete? We are faced with the same scenario as we move from weak to strong stability. As we mix weak and strong stability will we also see a transition from polynomal behaviour? And can a good definition of constrainedness be found for SMTI? Perhaps the most important future work is practical application, for example to the real-life hospital residents problem. To do so, it may be necessary to use encodings which take less time than $O(n^4)$ to establish consistency. We are actively examining one such encoding based on 0/1 variables which will take only $O(n^2)$ time: this should speed up run time per node while not necessarily reducing the number of search nodes.

ACKNOWLEDGEMENTS

We would like to thank Rob Irving, David Manlove, Barbara Smith, Francois Laburthe, Anton Morrison, our reviewers, and our stable partners Judith and Andrea.

REFERENCES

[1] P. Cheeseman, B. Kanefsky, and W Taylor, 'Where the really hard problems are', in *Proceedings IJCAI'91*, pp. 331–337, (1991).

[2] D. Gale and L.S. Shapley, 'College admissions and the stability of marriage', *American Mathematical Monthly*, **69**, 9–15, (1962).

[3] I.P. Gent, R.W. Irving, D.F. Manlove, P. Prosser, and B.M. Smith, 'A constraint programming approach to the stable marriage problem', in *Proceedings CP'01*, pp. 225–239, (2001).

[4] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh, 'The constrainedness of search', in *Proceedings AAAI'96*, pp. 246–252, (1996).

[5] D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, The MIT Press, 1989.

[6] K. Iwama, D.F. Manlove, S. Miyazaki, and Y. Morita, 'Stable marriage with incomplete lists and ties', in *Proceedings ICALP '99*, pp. 443–452, (1999).

[7] Francoise Laburthe. Choco: a constraint programming kernel for solving combinatorial optimization problems. http://www.choco-constraints.net/.

[8] A.K. Mackworth and E.C. Freuder, 'The complexity of some polynomial consistency algorithms for constraint satisfaction problems', *Artificial Intelligence*, **25**, 65–74, (1985).

[9] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, 'Hard variants of stable marriage', Technical Report TR-1999-43, Department of Computer Science, University of Glasgow, (1999).

[10] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, 'Hard variants of stable marriage', *to appear in Theoretical Computer Science*, (2002).

[11] J-C Regin, 'A filtering algorithm for constraints of difference in csps', in *Proceedings AAAI'94*, pp. 362–367, (1994).

[12] E.P.K. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, 1993.