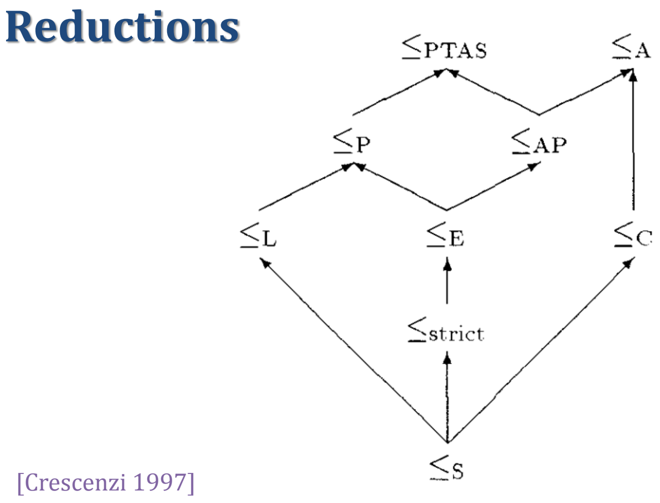
6.890: Algorithmic Lower Bounds: Fun With Hardness Proofs Fall 2014

Lecture 11 — October 9, 2014

Prof. Erik Demaine *Scribe: Nathan Pinsker, Patrick Yang, Rajan Udwani*

1 Recap

In the last lecture we looked at several different types of reductions for NPO problems as shown in the figure below.



Recall that one of the most commonly used reductions is the *L-reduction*, introduced by Papadimitriou and Yannakakis in [11]. This reduction consists of a pair of polynomial mappings $(f(.), g(.))$ where f maps an instance x of the problem A we are reducing from to an instance x' of the problem B we are reducing to, and g maps a feasible solution y' (of x' i.e. instance of B) to a feasible solution y (of x) such that the following conditions hold:

1. $OPT_B(x') \leq \alpha\ OPT_A(x)$
2. $|cost_A(y) - OPT_A(x)| \leq \beta\ |cost_B(y') - OPT_B(x')|$

for some non-zero α, β . Thus an L-reduction implies a PTAS-reduction with $\delta(\epsilon) = \epsilon/(\alpha\beta)$ for the minimization case. As the figure above shows L $\not\Rightarrow$ AP-reduction because in the maximization case $\delta(\epsilon) = \frac{1}{(\alpha\beta(1+1/\epsilon)-1)}$ which is a non-linear function.

Also, we have discussed the following APX-complete problems so far (results introduced in [3, 11]):

- **Max 3-SAT-3** and **Max3E-SAT-E5**: Where we L-reduced from Max 3-SAT and used properties of expander graphs to account for the bound on occurences of a variable.
- **Independent set** (for bounded degree graphs): By Strict reduction from the above problem

- **Vertex cover** (for bounded degree graphs): By L-reduction from the above since in bounded degree graphs both independednt set and vertex cover are $\Theta(|V|)$.
- **Dominating set** (bounded degree): By Strict redcution from Vertex cover.

2 Overview

In this lecture we will prove APX-completeness for Max-2SAT, Max NAE 3SAT, Max Cut and see a Schaefer-like exhaustive theorem that classifies a bunch of SAT related Max/Min problems and then another series of APX-completeness results in order to prove the same for the maximization version of the *edge matching* puzzle and then breifly mention some Log, Poly and Exp-APX completeness results and finally conclude by mentioning some problems in the classes NPO-complette and NPO-PB-complete (polynomially bounded costs).

3 Max 2SAT

We will do an L-reduction from the Independent set problem on bounded degree graphs by creating for each vertex v a clause with just the variable v and for each edge (v, w) a clause $\bar{v} \vee \bar{w}$. Now note that in any solution to the derived Max 2SAT instance, a certain edge clause is violated iff the 2 vertex clauses correponding to the endpoints of the edge are both true, but in such a case we can set any one of the 2 vertex variables to false and that will not decrease the number of satisfied clauses since the vertex clause corresponding to that vertex is dissatisfied but at least one new edge clause is satisfied and no other previously satisfied clause is changed. Hence we may assume in particular that in an optimal solution to the Max 2SAT instance all edge clauses are satisfied and hence we get a valid independent set by choosing the vertices corresponding to vertex variables set to zero. Note that $OPT_{2SAT} = OPT_{IS} + \#$ edges, and since $OPT_{IS} = \Theta(|V|)$ and $\#$ edges$= \Theta(|V|)$ the reduction satisfies condition (1) of L-reduction. Also condition (2) is satisfied for $\beta = 1$ because the additive term # edges cancels off.

With some more work one can show a similar result for Max E2SAT-E3 as well.

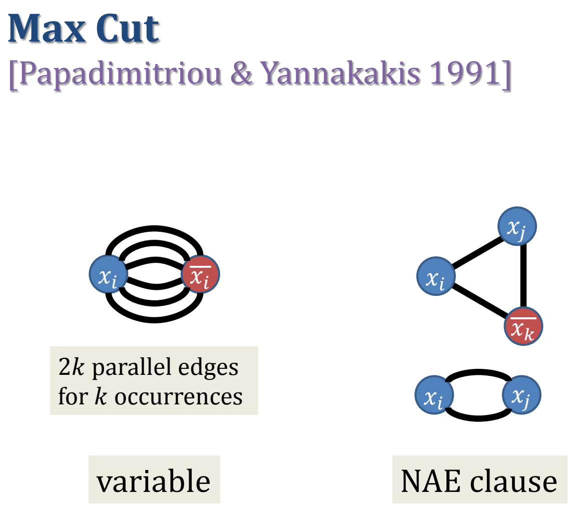
4 Max NAE 3SAT

We do a strict reduction from Max 2SAT by creating a NAE 3SAT clause (x, y, a) for a 2SAT clause $x \vee y$ (and NAE (x, x, a) for a one variable clause). Note that the extra variable a is global and common amongst all the NAE clauses created. Now, if in an assignment to the NAE problem a is set to 1, we can flip all variables and thus get $a = 0$ without violating any NAE clause. Therefore, we may assume $a = 0$. Hence for the same assignment S to variables and $a = 0$, $OPT_{NAE} = OPT_{2SAT}$ and $cost_{NAE}(S) = cost_{2SAT}(S)$.

5 Max Cut

Note that the reduction below also proves APX-completeness for Max Positive 1-in-2SAT and Max Positive XOR SAT because of their similarity to graph cuts (any cut edge has each endpoint in a different vertex set).

We will do an L-reduction from Max NAE 3SAT. The figure below shows the variable and clause



gadgets.

Notice that in the clause gadget either all vertices are in the same cut (all equal) or one is in a different cut (not all equal) and thus we get a cut contribution of either 0 or 2. In every variable gadget, to ensure that the true and false versions of a variable are not in the same cut we can add $2k$ parallel edges between, where k is # occurences of the variable and with this we can assume that in any solution all variable gadgets are split up, because otherwise we can split them up and get an increase of $2k$ in the cut and since by doing this at most k clause gadgets might end up entirely in one cut, we lose at most $2k$ in terms of cut value, and thus the net cut value increases.

Hence, $OPT_{Cut} = 2(\sum_i \# \; occurences \; of \; x_i + \# \; satisfied \; clauses)$ and since total # occurences of variable is $\leq 3(\#$ clauses), we get that $OPT_{Cut} = \Theta(OPT_{NAE})$ (for $OPT_{NAE} \geq 0.5(\#\text{clauses})$). Once again the additive term corresponding to occurence of variable cancels off and we get (2) as well.

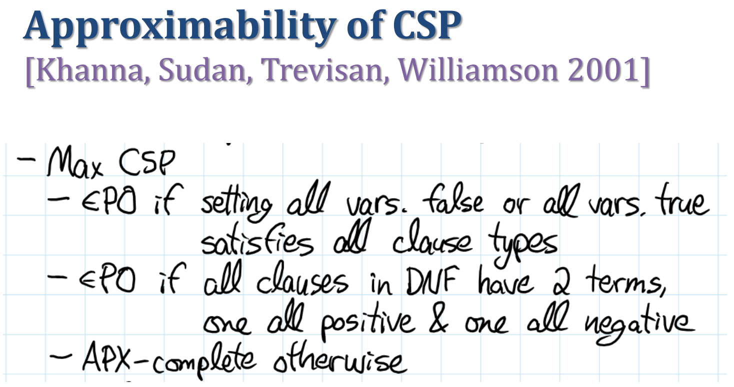
6 Max/Min CSP/Ones

This theorem (introduced by Khanna, Sudan, Trevisan, and Williamson in [10]) is analogous to Schaefer's dichotomy theorem for hardness of SAT variants. However, it instead classifies optimization versions of SAT or more generally Constraint Satisfaction Problems (CSP). For instance, Max Cut can be written as Max CSP (x_i XOR x_j, $\forall(i,j) \in E$), also clauses and variables may have distinct weights. 'CSP' problems are ones in which the objective is to optimize the number of clauses satisfied. 'Ones' refers to the problem of optimizing the number of variables that are 1 in a satisfying assignment.

Unlike Schaefer's theorem, however, there is no simple or intuitive description of the dichotomy in each of the four cases. Thus, we instead provide the full listings of subcases from the lecture slides. Do not memorize these lists! Instead, refer back to them if your problem seems to have CSP characteristics. In these slides, **read the subcases sequentially**. In the below sections, PO refers to a complexity class of problems optimizable in polynomial time.

All of the below cases work when the clauses (for CSP) or variables (for Ones) are weighted, EXCEPT FOR the one highlighted case in Min Ones.

6.1 Max CSP

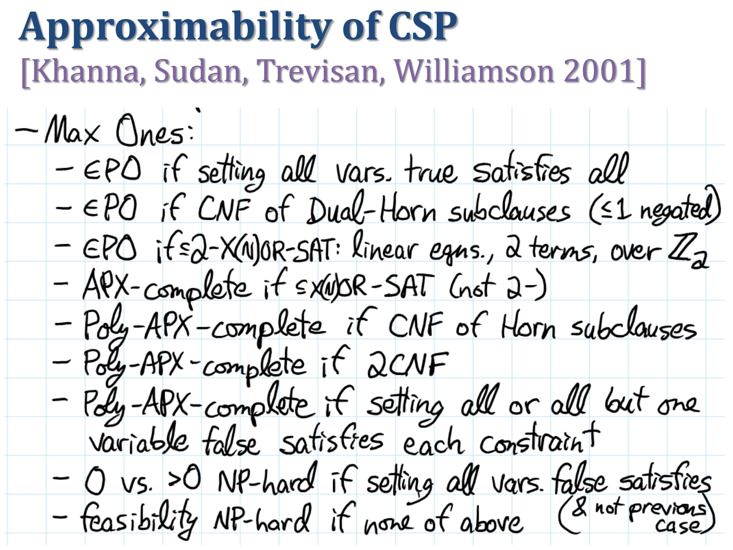


Approximability of CSP

[Khanna, Sudan, Trevisan, Williamson 2001]

- Max CSP
  - $\in$PO if setting all vars. false or all vars. true satisfies all clause types
  - $\in$PO if all clauses in DNF have 2 terms, one all positive & one all negative
  - APX-complete otherwise

6.2 Max Ones



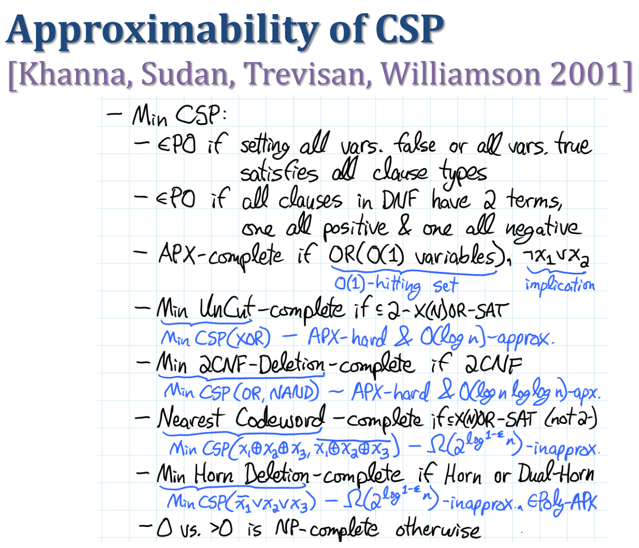
Approximability of CSP

[Khanna, Sudan, Trevisan, Williamson 2001]

- Max Ones:
  - $\in$PO if setting all vars. true satisfies all
  - $\in$PO if CNF of Dual-Horn subclauses (≤ 1 negated)
  - $\in$PO if $\leq$2-X(N)OR-SAT: linear eqns., 2 terms, over $\mathbb{Z}_2$
  - APX-complete if $\leq$X(N)OR-SAT (not 2-)
  - Poly-APX-complete if CNF of Horn subclauses
  - Poly-APX-complete if 2CNF
  - Poly-APX-complete if setting all or all but one variable false satisfies each constraint
  - 0 vs. >0 NP-hard if setting all vars. false satisfies
  - feasibility NP-hard if none of above (& not previous case)

Note that if setting all variables to False satisfies the problem, then Max Ones need not be in PO; in fact, it is hard. This is because finding **any** other feasible solution may be NP-hard. This scenario is listed as ”0 vs > 0”.

6.3 Min CSP



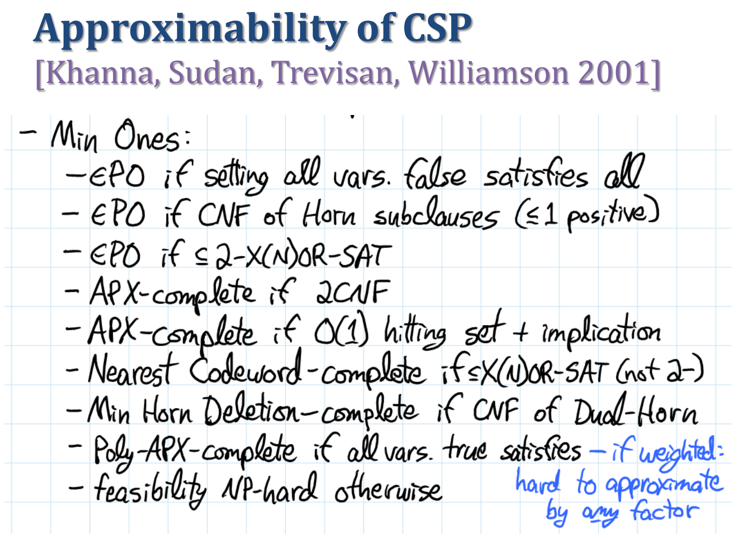
Approximability of CSP

[Khanna, Sudan, Trevisan, Williamson 2001]

- Min CSP:
  - $\in$PO if setting all vars. false or all vars. true satisfies all clause types
  - $\in$PO if all clauses in DNF have 2 terms, one all positive & one all negative
  - APX-complete if $\underbrace{\text{OR}(O(1)\text{ variables})}_{O(1)\text{-hitting set}}$, $\underbrace{\neg x_1 \vee x_2}_{\text{implication}}$
  - $\underbrace{\text{Min UnCut}}$-complete if $\leq$2-X(N)OR-SAT
    Min CSP(XOR) — APX-hard & $O(\log n)$-approx.
  - $\underbrace{\text{Min 2CNF-Deletion}}$-complete if 2CNF
    Min CSP (OR, NAND) ~ APX-hard & $O(\log n \log\log n)$-apx.
  - $\underbrace{\text{Nearest Codeword}}$-complete if $\leq$X(N)OR-SAT (not 2-)
    Min CSP$(x_1 \oplus x_2 \oplus x_3, \overline{x_1 \oplus x_2 \oplus x_3})$ — $\Omega(2^{\log^{1-\varepsilon} n})$-inapprox.
  - $\underbrace{\text{Min Horn Deletion}}$-complete if Horn or Dual-Horn
    Min CSP$(\bar{x}_1 \vee x_2 \vee x_3)$ — $\Omega(2^{\log^{1-\varepsilon} n})$-inapprox., $\in$Poly-APX
  - 0 vs. >0 is NP-complete otherwise

The fourth through seventh subcases listed in the above slide are classes corresponding to more familiar problems, namely UnCut, 2CNF-Deletion, Nearest Codeword, and Horn Deletion.

6.4 Min Ones



Approximability of CSP

[Khanna, Sudan, Trevisan, Williamson 2001]

- Min Ones:
  - $\in$PO if setting all vars. false satisfies all
  - $\in$PO if CNF of Horn subclauses ($\leq$1 positive)
  - $\in$PO if $\leq$2-X(N)OR-SAT
  - APX-complete if 2CNF
  - APX-complete if O(1) hitting set + implication
  - Nearest Codeword-complete if $\leq$X(N)OR-SAT (not 2-)
  - Min Horn Deletion-complete if CNF of Dual-Horn
  - Poly-APX-complete if all vars. true satisfies — if weighted: hard to approximate by any factor
  - feasibility NP-hard otherwise

7 Edge Matching Approximation

We return to a familiar problem from early in the course: edge matching puzzles. Instead of trying to match every single edge, however, we now attempt to maximize the number of edges that are matched when all pieces are placed. We reduce through Max Independent Set on 3-regular 3-edge-colorable graphs.

7.1 Max Independent Set and Max 3DM-E2

In this section, for the purposes of reducing to edge matching, we consider Max Independent Set on 3-regular 3-edge-colorable graphs. This approximation problem is APX-Complete. We do not show the proof here; it is due to Chlebik and Chlebikova [5].

Note in particular the characteristics of a 3-edge-coloring on a 3-regular graph. Each vertex must have one edge of each color.

Now consider another variant on a familiar problem: Max 3DM-E2. Recall that 3DM is the 3-dimensional matching problem presented earlier during our course. E2 refers to each element in the 3DM problem appearing exactly twice in selectable triples. We show a strict reduction from the Max Independent Set variant above.

Consider a 3-edge coloring on a 3-regular graph. Let the edges be the elements of a 3-dimensional matching problem, such that the three classes in the 3-dimensional matching problem correspond to the three colors on the graph. Let triples on the edges correspond to vertices - as we noted before, each vertex has one edge of each color, so it will correspond to a triple of elements in 3-dimensional matching from different classes. Each edge has two endpoints, so it is in exactly two triples in 3DM. Two selected vertices that are independent correspond to two selected triples in the 3DM problem that do not share any elements. Thus we have a strict reduction. (In fact this is an equivalence, as we can reconstruct a 3-regular 3-colorable graph from the 3DM-E2 instance and perform the reverse reduction).

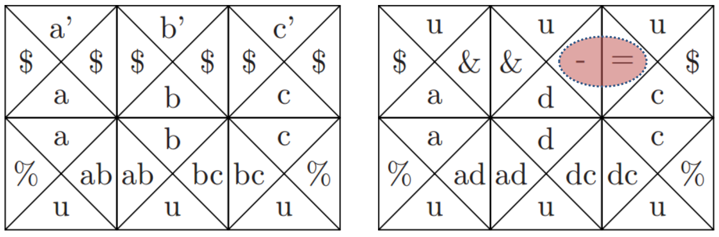
7.2 Max Edge Matching

First of all, the problem is in APX. A maximum matching algorithm (as in, one that just attempts to find the maximum number of pairs of tiles that share an edge) will give a $n/8$ approximation just by matching one pair of edges per 8 edges.

Now we show that Max Edge Matching is APX-Hard, and thus APX-Complete. The proof is now a reduction from Max 3DM, due to Antoniadis and Lingas [1]. We do not cover the details of the proof in this lecture, but we will note some particular characteristics.

Edge Matching Puzzles

[Antoniadis & Lingas 2010]



picked triple (a, b, c)

unpicked triple (a, d, c)

Above are the gadgets for a picked triple and unpicked triple in an edge matching puzzle. The row of us on the bottom is intended to match up against an unpictured boundary structure in a way that forces the placing of the bottom row (in the sense that placing the bottom row in the intended way can only improve the number of matches).

Note that the tile marked with two dollar signs, a, and a' is unique to the letter a, which only allows us to pick one triple containing a. We also need structures that enable us to dump the 'picked' top row if we do not pick the triple, or the 'unpicked' top row if we do pick the triple. The dump structures are listed, together with a full listing of tiles per 3DM element, in the lecture slides; we omit them here for brevity.

8 Other Complexity Classes for Approximations

APX-Completeness is not the end-all be-all of complexity in approximations. There exists a wealth of classes both above it and below it in complexity.

First of all, there exist problems in APX that have no PTAS but are not APX-complete, assuming P $\neq$ NP. Such problems are called 'APX-intermediate'; this is not a complexity class but a categorization o problems [6]. Familiar APX-intermediate problems include:

- Bin Packing - we can always find a $(1 + \epsilon) \cdot \text{OPT} + 1$ approximation (this may be familiar to those of you who have taken Advanced Algorithms). This is not a PTAS due to the +1 additive factor (it is an 'asymptotic' PTAS), but nevertheless it is an approximation that can be done in polynomial time.
- Minimizing the max degree of a spanning tree
- Minimum edge coloring

There exist problems that cannot be efficiently approximated within a constant factor. We provide a brief overview of some of the more commonly studied "harder" complexity classes.

8.1 Log-APX-completeness

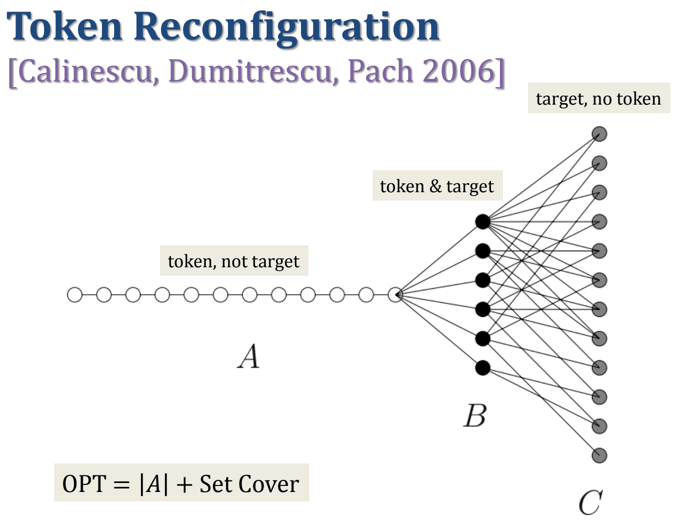
Optimal solutions to problems in this class are only approximable to within a logarithmic factor. Two of the most famous problems in this class are set cover and dominating set [7], each of which can be reduced to the other without much trouble:

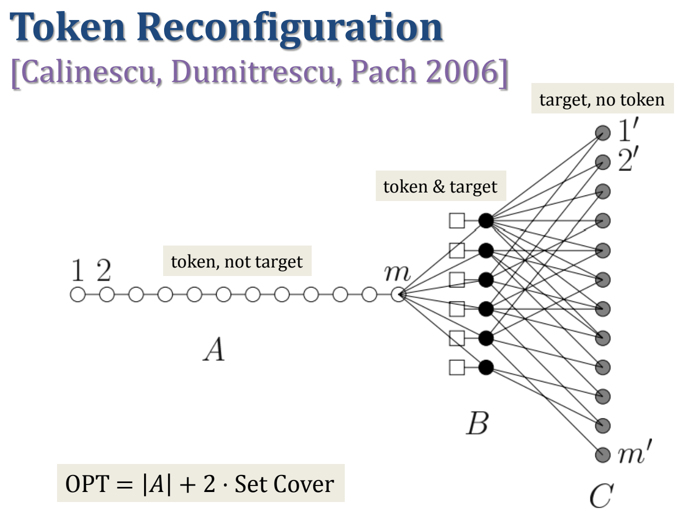
- Given an instance G of a dominating set problem, we create a set S_v for each vertex $v \in G$, which contains v as well as all its neighbors. A set $\{S_{a_i}\}$ of these sets is a cover of G exactly when the set $\{a_i\}$ of vertices is a set cover.
- Given an instance S of a set cover problem, we create a vertex for each set in S as well as each element. We connect all vertices representing sets to each other, so they form a clique. We then connect the vertex representing set S_i to each vertex representing element j iff set S_i contains j. It is always better to choose vertices representing sets, and so a valid dominating set is exactly a set cover of S.

8.2 Token reconfiguration

In this problem, we are given a graph G, each of whose vertices may contain a robot. We are given an initial configuration of the robots (vertices that start with robots are called "tokens"), and want a final configuration with robots in "target" vertices. One move consists of moving a robot along an arbitrary path that does not contain any other robots, and we wish to minimize the number of moves. We will reduce this problem to set cover using an L-reduction, to show that it is APX-complete [4].

The motivation for this problem comes from the 15 (or $n^2 - 1$ for some n) puzzle, which is known to be NP-Hard and in APX, but it's unknown whether it's APX-complete [12].





Given an instance of the token reconfiguration problem of this form, the optimal solution takes a number of moves equal to $|A|$ plus the size of a minimal set cover. This is because we need one move to fill every location in C, which requires exactly $|A|$ moves. Furthermore, we need one move to cover every location vacated by a robot in B. But the number of robots in B that must move is exactly the size of the minimum set cover.

Thus, the conditions of the L-reduction are satisfied, and this problem is NP-complete. If two robots are not allowed to occupy the same vertex at once, the second image shows a small modification that can be used to get around this problem. The optimal solution in this case is $|A|$ plus twice the size of a minimal set cover.

8.3 Poly-APX-complete

Optimal solutions to problems in this class are approximable to within a polynomial factor. Two problems in this class (each of which can easily be reduced to the other) are finding the largest clique and the largest independent set [2].

8.4 Exp-APX-complete

Encountering this class is rather uncommon. Exp-APX-complete problems occur when there are numbers in the problem that can have large size, increasing the difficulty of approximation. The most well-known problem in this class is the non-metric traveling salesman problem, where edges can be exponential in the size of the input [8].

8.5 NPO-complete

This class contains solutions of polynomial size, whose decision problems are NP-complete [6]. Examples of problems in this class include:

- maximum/minimum weighted Ones (maximize number of true variables in clauses)
- maximum/minimum 0-1 linear programming

8.6 NPOPB-complete

This class contains solutions that are recognizable in polynomial time, whose costs are also computable in polynomial time. (The PB in the title stands for "polynomially bounded".) However, computing even a valid solution is NP-complete. The distinction between NPOPB-complete and NPO-complete is similar to that between Poly-APX-complete and Exp-APX-complete respectively; it's usually due to the presence of large numbers.

NPOPB-complete problems are difficult to approximate polynomially even when the solutions are trivial [9]. Examples of problems in this class include:

- minimum independent dominating set
- shortest Turing-machine computation
- longest induced path
- longest path with "forbidden pairs" (pairs of vertices such that the path cannot cross both)

References

[1] Antonios Antoniadis and Andrzej Lingas. Approximability of edge matching puzzles. In *SOFSEM 2010: Theory and Practice of Computer Science, 36th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 23-29, 2010. Proceedings*, pages 153–164, 2010.

[2] Cristina Bazgan, Bruno Escoffier, and Vangelis Th. Paschos. Completeness in standard and differential approximation classes: Poly-(d)apx- and (d)ptas-completeness. *Theor. Comput. Sci.*, 339(2-3):272–292, 2005.

[3] Piotr Berman and Marek Karpinski. On some tighter inapproximability results (extended abstract). In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, pages 200–209, 1999.

[4] Gruia Călinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. *SIAM J. Discrete Math.*, 22(1):124–138, 2008.

[5] Miroslav Chlebík and Janka Chlebíková. Approximation hardness for small occurrence instances of np-hard problems. In *Algorithms and Complexity, 5th Italian Conference, CIAC 2003, Rome, Italy, May 28-30, 2003, Proceedings*, pages 152–164, 2003.

[6] Pierluigi Crescenzi, Viggo Kann, Riccardo Silvestri, and Luca Trevisan. Structure in approximation classes. *SIAM J. Comput.*, 28(5):1759–1782, 1999.

[7] Bruno Escoffier and Vangelis Th. Paschos. Completeness in approximation classes beyond APX. *Theor. Comput. Sci.*, 359(1-3):369–377, 2006.

[8] Bruno Escoffier and Vangelis Th. Paschos. Completeness in approximation classes beyond APX. *Theor. Comput. Sci.*, 359(1-3):369–377, 2006.

[9] Peter Jonsson. Near-optimal nonapproximability results for some NPO pb-complete problems. *Inf. Process. Lett.*, 68(5):249–253, 1998.

[10] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, December 2001.

[11] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

[12] Daniel Ratner and Manfred Warmuth. The (¡ i¿ n¡/i¿¡ sup¿ 2¡/sup¿- 1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.