Towards Ambitious Approximation Algorithms in Stubborn Set Optimization

Kimmo Varpaaniemi [*]
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O. Box 9205, FIN-02015 HUT, Finland
kimmo.varpaaniemi@hut.fi

Abstract. This paper continues research on the stubborn set method that constructs on-the-fly a reduced LTS that is CFFD-equivalent to the parallel composition of given LTSs. In particular, minimization of the number of successor states of a given state is reconsidered. The earlier suggested and/or-graph approach requires solving #P-complete counting problems in order to get the weights for the vertices of the and/or-graph. The "branch-and-bound" decision problem corresponding to the minimization of the sum of the computed weights is "only" NP-complete. Unfortunately, #P-complete counting does not seem easily avoidable in the general case because it is PP-complete to check whether a given stubborn set produces at most as many successor states as another given stubborn set. Instead of solving each of the subproblems, one could think of computing approximate solutions in such a way that the total effect of the approximations is a useful approximation itself. General approximation algorithms for propositional logic problems essentially suffice for the purpose.

Keywords: LTSs, stubborn sets, and/or-graphs, approximability in optimization and counting

1 Introduction

In process-algebraic approaches to formal verification, it is usual to describe behaviors of actual systems by defining *LTSs* (*labelled transition systems*) and operations which produce LTSs from LTSs. For example, there are several ways to define a *parallel composition* of LTSs. If correspondence between actions and atomic propositions is defined appropriately, *CFFD-equivalence* (*chaos free failures divergences equivalence*) guarantees preservation of satisfiability/unsatisfiability of any linear time temporal logic formula that does not contain

[*]This work has been funded by the Academy of Finland (Project 47754).

the next-time operator. More information about equivalences and preorders related to CFFD-equivalence can be found e.g. from [8, 19].

Combinatorial explosion tends to occur in parallel compositions, but the explosion can be alleviated by eliminating redundant interleavings. The *stubborn set method* is able to do such elimination by constructing on-the-fly a reduced LTS that is CFFD-equivalent to the actual parallel composition [17, 18]. It is a system-independent and intuitively appealing heuristic to minimize the number of successor states of each state in the reduced LTS though it is well known that the resulting reduced LTS itself is not necessarily minimal w.r.t. the number of states.

This paper continues the research done in [21]. The number of successor states of a given state can be minimized by constructing an and/or-graph with weighted vertices and by finding a set of vertices that satisfies a certain constraint such that no set of vertices satisfying the constraint has a smaller sum of weights. The minimization can be done in a branch-and-bound manner, by repeatedly solving instances of the corresponding decision problem which is NP-complete for a given weighted and/or-graph. The number of the instances is at most logarithmic w.r.t. the sum of weights of all vertices.

Without weights, the and/or-graph can be constructed in low-degree polynomial time. However, since actions can be nondeterministic and transitions can share target states, it is by no means obvious how long it takes to compute the weights. It turns out to be #P-complete to compute a weight for a vertex in a nontrivial case. We also get to know that it is PP-complete to check whether a given stubborn set produces at most as many successor states as another given stubborn set. It thus seems unlikely that #P-complete counting could generally be avoided by using some other approach.

All these complexity results motivate us to seek approximate rather than precise solutions. An obvious approximation is to carry out the above mentioned branch-and-bound search only up to a small extent, the extreme being: "If some stubborn set produces only one successor state, find such a set. Otherwise choose the set of all actions or any other known stubborn set." Such compromises are successful in many applications, and it is sometimes tempting to speculate how much of the practical value of the method is due to those stubborn sets which do not cause branching. However, there should be room for more ambitious approximation algorithms. Moreover, it should be possible to compute approximate solutions to subproblems in such a way that the total effect of the approximations is a useful approximation itself. General approximation algorithms for propositional logic problems essentially suffice for the purpose because each of the difficult subproblems of our minimization problem is an instance of or very close to some generic propositional logic problem.

The rest of the paper has been structured as follows. Some "complete" propositional logic problems are mentioned in Section 2. And/or-graphs are considered in Section 3, LTSs in Section 4, and stubborn sets in Section 5. We shall use N to denote the set of nonnegative integer numbers. For any $a \in N$, $N_{<a} = \{i \in N \mid i < a\}$.

2 Reference problems from propositional logic

This paper assumes that introduction to complexity theory and approximability theory is available, e.g. from [1, 13]. On the other hand, it is built in the concept of completeness that each of the below "complete" problems in a sense characterizes the associated complexity class. Each of the below problems is mentioned in some of the later sections.

SAT: "Given a propositional formula in a conjunctive normal form, check whether the formula is satisfiable." SAT is NP-complete [2].

WSAT: "Given a propositional formula in a conjunctive normal form and nonnegative integer weights for the variables, find a total assignment that satisfies the formula (when satisfiable) and minimizes the sum of weights over variables having the value true." WSAT is NPO-complete [12].

W2SAT: the special case of WSAT where no conjunct contains more than two literals. W2SAT is MAX-SNP-complete [6].

DNFCOUNT: "Given a propositional formula in a disjunctive normal form, compute the number of total assignments that satisfy the formula." DNFCOUNT is #P-complete [16].

CNFENOUGH: "Given an integer number k and a propositional formula in a conjunctive normal form, check whether at least k total assignments satisfy the formula." CNFENOUGH is PP-complete [15].

3 And/or-graphs

Definition 3.1 An *and/or-graph* is a 4-tuple $\langle V_\otimes, V_\oplus, \kappa, F\rangle$ such that $V_\otimes$ is the set of *and-vertices*, $V_\oplus$ is the set of *or-vertices*, $V_\otimes \cap V_\oplus = \emptyset$, $V_\otimes \cup V_\oplus$ is finite, $\kappa \in V_\otimes \cup V_\oplus$, $F \subseteq (V_\otimes \cup V_\oplus) \times (V_\otimes \cup V_\oplus)$ is the set of *edges*, and $\forall y \in V_\oplus : \ F(y) \neq \emptyset$.
A set $L \subseteq V_\otimes \cup V_\oplus$ is *legal* iff
$(\kappa \in L) \wedge (\forall x \in V_\otimes \cap L : \ F(x) \subseteq L) \wedge (\forall y \in V_\oplus \cap L : \ F(y) \cap L \neq \emptyset)$. For any subsets L, H and P of $V_\otimes \cup V_\oplus$, L is a *H-solidity witness* of P iff L is legal and $P = H \cap L$. For any $H \subseteq V_\otimes \cup V_\oplus$, a set $P \subseteq V_\otimes \cup V_\oplus$ is *H-solid* iff some subset of $V_\otimes \cup V_\oplus$ is a H-solidity witness of H. □

The contents of Definition 3.1 and the below two problem formulations are from [21].

SOLIDWEIGHT: "Given $n \in N$, an and/or-graph, a subset H of its vertices and a function w from H to N, is there an H-solid set P such that $\sum_{p \in P} w(p) \leq n$?"

SOLIDOPTW: "Given an and/or-graph, a subset H of its vertices and a function w from H to N, find an H-solid set A such that for each H-solid set P, $\sum_{a \in A} w(a) \leq \sum_{p \in P} w(p)$."

Clearly, SOLIDWEIGHT is the "branch-and-bound" decision problem corresponding to SOLIDOPTW. NP-completeness of SOLIDWEIGHT is essentially shown in [14] though also "confirmed" by [20, 21]. SOLIDOPTW is a special case of WSAT. By looking at the definition of legal sets, it is easy to write such a formula for SOLIDOPTW in a conjunctive normal form in such a way that no variable occurs negated in more than one conjunct, no conjunct contains more than one negative literal, and each conjunct contains at least one positive literal. (If the signs of the literals were reversed, every conjunct would be a Horn clause.)

So, approximation algorithms for WSAT are applicable to SOLIDOPTW. Unfortunately, NPO-completeness of WSAT [12] and the related unapproximability results [1, 7, 9] indicate that every polynomial time approximation algorithm produces relatively poor approximations for at least some instances of WSAT. What comes to the possibility to reduce WSAT to SOLIDOPTW, it seems that a polynomial time reduction can be obtained by revising the the reduction from SAT to SOLIDWEIGHT (essentially presented in [14]) w.r.t. weights, but the obtained reduction does not support approximation by taking preimages. (There are solid sets that do not have satisfying total assignments as preimages w.r.t. the reduction.) So, the complexity and approximability of SOLIDOPTW are unclear to some extent.

Anyway, it should be kept in mind that complexity and approximability results are worst-case results and often do not tell much about the situation in some special case. For example, W2SAT has polynomial time algorithms guaranteeing the sum of weights in the approximate solution to be at most twice the minimum value when the formula is satisfiable [1, 4, 5, 6]. MAX-SNP-completeness of W2SAT [6] may or may not explain why no better algorithm is known. It would be interesting to know what kind of instances of SOLIDOPTW have an approximation supporting polynomial time reduction to W2SAT. At present, we do not exclude the possibility that all instances would have such a reduction. Nevertheless, there should be more such instances than those that are trivially seen from the above specified formula for SOLIDOPTW. Namely, if that formula has at most two literals in each conjunct, then every or-vertex (if any) in the and/or-graph has only one outgoing edge, and a unique solution to SOLIDOPTW is trivially determined by the set of vertices accessible from κ.

4 LTSs

Definition 4.1 A *labelled transition system* (an *LTS*) is a 5-tuple $L = \langle S, \mho, \Im, \Delta, \iota \rangle$ such that S is the set of *states*, $\mho$ is the set of *visible actions*, $\Im$ is the set of *invisible actions*, $\mho \cap \Im = \emptyset = S \cap (\mho \cup \Im)$, $\Delta \subseteq S \times (\mho \cup \Im) \times S$ is the set of *transitions*, and $\iota \in S$ is the *initial state.* For each transition $\langle x, a, y \rangle$, a is the *label of the transition* and y is the *target state of the transition.* A transition $\langle x, a, y \rangle$ is a *successor transition* of a state z iff $z = x$. A state y is a *successor state* of a state x iff there is some action a such that $\langle x, a, y \rangle$ is a transition. L is a *finite LTS* iff $S \cup \mho \cup \Im$ is finite. An action a is *enabled at a state* x iff there is a state y such that $\langle x, a, y \rangle \in \Delta$. The set of actions enabled at a state x is denoted by $\eta(x)$. □

Definition 4.2 Let $n \in N \setminus \{0,1\}$. For each $i \in N_{<n}$, let $L_i = \langle S_i, \mho_i, \Im_i, \Delta_i, \iota_i \rangle$ be an LTS. The n-tuple $\langle L_0, \ldots, L_{n-1} \rangle$ is *action-consistent* iff for all different i and r in $N_{<n}$, $\mho_i \cap \Im_r = \emptyset$ and $(S_0 \times \ldots \times S_{n-1}) \cap (\mho_i \cup \Im_i) = \emptyset$. If $\langle L_0, \ldots, L_{n-1} \rangle$ is action-consistent, the *parallel composition* of $\langle L_0, \ldots, L_{n-1} \rangle$ is the LTS $L = \langle S, \mho, \Im, \Delta, \iota \rangle$ such that $S = S_0 \times \ldots \times S_{n-1}$, $\mho = \cup_{i=0}^{n-1} \mho_i$, $\Im = \cup_{i=0}^{n-1} \Im_i$, $\iota = \langle \iota_0, \ldots, \iota_{n-1} \rangle$, for any $a \in \mho \cup \Im$, for any $x = \langle x_0, \ldots, x_{n-1} \rangle \in S$ and for any $y = \langle y_0, \ldots, y_{n-1} \rangle \in S$, $\langle x, a, y \rangle \in \Delta$ iff $\forall i \in N_{<n} : \ ((a \in \mho_i \cup \Im_i) \wedge (\langle x_i, a, y_i \rangle \in \Delta_i)) \vee ((a \notin \mho_i \cup \Im_i) \wedge x_i = y_i)$. □

The contents of Definitions 4.1 and 4.2 are from [21]. The parallel composition operation can be thought to carry out hiding after fine-grained alphabet-based synchronization.

Proposition 4.3 *It is #P-complete to compute the number of successor states of a given state in the parallel composition of a given action-consistent tuple of finite LTSs.*

Proof. Inclusion in #P is obvious because it is easy to design a nondeterministic Turing machine which is able to produce any syntactically possible candidate successor state and then check in polynomial time whether the produced candidate is really a successor state. #P-hardness follows by the following polynomial time reduction from DNFCOUNT.

Let m and n be in $N \setminus \{0,1\}$. Let $\{y_0, \ldots, y_{n-1}\}$ be the set of variables and $D = C_0 \vee \ldots \vee C_{m-1}$ the formula in disjunctive formal form. For each $i \in N_{<n}$ and for each $j \in N_{<m}$, we define the set $A_{i,j}$ as follows. If both y_i and its negation occurs in C_j, then $A_{i,j}$ is empty. If y_i occurs only positively in C_j, then $A_{i,j} = \{1\}$. If y_i occurs only negatively in C_j, then $A_{i,j} = \{0\}$. If y_i does not occur in any form in C_j, then $A_{i,j} = \{0,1\}$.

For each $i \in N_{<n}$, let $L_i = \langle S_i, \mho_i, \Im_i, \Delta_i, \iota_i \rangle$ be an LTS such that $S_i = \{0,1,2\}$, $\iota_i = \{2\}$, $\mho_i = \emptyset$, $\Im_i = N_{<m}$, and $\Delta_i = \cup_{j=0}^{m-1} \{2\} \times \{j\} \times A_{i,j}$.

Now $\langle L_0, \ldots, L_{n-1} \rangle$ is action-consistent. For the initial state in the parallel composition of $\langle L_0, \ldots, L_{n-1} \rangle$, for each $j \in N_{<m}$, the number of successor transitions labelled by j is equal to the number of total assignments satisfying C_j. Consequently, the number of successor states of the initial state is equal to the number of total assignments satisfying D. □

In [21], every vertex weight in the and/or-graphs for the "minimize the number of successor states" problems is actually the number of successor states of a certain state in the parallel composition of a tuple of certain LTSs. Proposition 4.3 indicates that we should (again) seek approximate rather than precise solutions. Due to the very direct reduction in the above proof, the Monte-Carlo approximation approach presented by [10, 11] for DNFCOUNT can easily be revised to approximate the number of successor states. What is really important is that by taking only "tolerably" many samples in the Monte-Carlo method, almost as good approximate solutions as wanted can be obtained, at least statistically. So, it is possible to constrain the precision of not only the approximate solutions but also their sums.

5 Stubborn sets

Repeating the style of [21], we assume the following.

- $n \in N \setminus \{0, 1\}$, and for each $i \in N_{<n}$, $L_i = \langle S_i, \mho_i, \Im_i, \Delta_i, \iota_i \rangle$ is a finite LTS, and η_i is the "η of L_i" w.r.t. Definition 4.1.
- $\langle L_0, \ldots, L_{n-1} \rangle$ is action-consistent, $L = \langle S, \mho, \Im, \Delta, \iota \rangle$ is the parallel composition of $\langle L_0, \ldots, L_{n-1} \rangle$, and η_L is the "η of L" w.r.t. Definition 4.1.
- $s = \langle s_0, \ldots, s_{n-1} \rangle \in S$, and at least one invisible action is enabled at s.

Definition 5.1 Let $A \subseteq \mho \cup \Im$ and $X \subseteq S$. A is *CFFD-stubborn* at $\langle s, X \rangle$ the following conditions hold.

- $\Im \cap \eta_L(s) \cap A \neq \emptyset$. In other words, A contains at least one invisible action that is enabled at s.
- $\mho \cap \eta_L(s) \cap A = \emptyset$ or $\mho \subseteq A$. In other words, if A contains some visible action that is enabled at s, then A contains all visible actions.
- For each $a \in A \cap \eta_L(s)$ and for each $i \in N_{<n}$, $(a \notin \mho_i \cup \Im_i) \vee (\eta_i(s_i) \subseteq A)$.
- For each $a \in A \setminus \eta_L(s)$, there is some $i \in N_{<n}$ such that $a \in (\mho_i \cup \Im_i) \setminus \eta_i(s_i)$ and $\eta_i(s_i) \subseteq A$.
- $(\{s\} \times (\Im \cap A) \times (X \cup \{s\})) \cap \Delta = \emptyset$ or $\mho \subseteq A$. In other words, if some successor transition of s leads to some state of $X \cup \{s\}$ and the label of the transition is an invisible action of A, then A contains all visible actions. □

Definition 5.2 Let $A \subseteq \mho \cup \Im$, $Q \subseteq \mho \cup \Im$, and $X \subseteq S$. Q is a *CFFD-eligibility witness* of A at $\langle s, X \rangle$ iff Q is CFFD-stubborn at $\langle s, X \rangle$ and $A = Q \cap \eta_L(s)$. A is *CFFD-eligible* at $\langle s, X \rangle$ iff some subset of $\mho \cup \Im$ is a CFFD-eligibility witness of A at $\langle s, X \rangle$. □

The contents of Definitions 5.1 and 5.2 are from [21] where the definition of CFFD-equivalence and the connection between CFFD-stubbornness and CFFD-equivalence is given. We now recall one of the optimization problems of [21] and give a "naive sub-problem" for it.

CFFDELIGSTOPT: "Given $X \subseteq S$, find a CFFD-eligible set A at $\langle s, X \rangle$ such that the number of successor states of s via transitions labelled by actions of A is the least possible."

CFFD2ELIG: "Given $X \subseteq S$ and CFFD-eligible sets Q and R at $\langle s, X \rangle$, is the number of successor states of s via transitions labelled by actions of Q less than or equal to the number of successor states of s via transitions labelled by actions of R?"

Conforming to [21], we assume the following.

- In the below expressions, p is the cardinality of the alphabet for encoding the input, $\alpha_i = \lceil \log_p(1 + |S_i|) \rceil$, and $\beta_i = \lceil \log_p(1 + |\mho_i \cup \Im_i|) \rceil$.
- The length of the input of CFFDELIGSTOPT is at least $\sum_{i=0}^{n-1} (\alpha_i + |X|\alpha_i + |(\{s_i\} \times (\mho_i \cup \Im_i) \times S_i) \cap \Delta_i|\alpha_i + |\mho_i \cup \Im_i|\beta_i)$.
- The length of the input of CFFD2ELIG is at least $(|Q| + |R|) \lceil \log_p(1 + |\mho \cup \Im|) \rceil$ plus the length of the input of CFFDELIGSTOPT.

Lemma 5.3 *CFFD2ELIG is PP-hard.*

Proof. We get a polynomial time reduction from CNFENOUGH. Let k be an integer number and m and n in $N \setminus \{0, 1\}$. Let $\{y_0, \ldots, y_{n-1}\}$ be the set of variables and $E = F_0 \wedge \ldots \wedge F_{m-1}$ the formula in conjunctive normal form. If $k < 1$, it suffices to construct any positive-answer instance of CFFD2ELIG. If $k > 2^n$, it suffices to construct any negative-answer instance of CFFD2ELIG. Let thus $1 \leq k \leq 2^n$. Let $D = C_0 \vee \ldots \vee C_{m-1}$ be the negation of F converted in a disjunctive normal form by using De Morgan's law.

As shown in [3], it is possible to construct in low-degree polynomial time for n variables a formula G in disjunctive normal form (allowing contradictions) with at least two disjuncts such that the number of total assignments satisfying G is $2^n - k$. Let thus $G = C_m \vee \ldots \vee C_{m+r-1}$ be such a formula for new variables $\{y_n, \ldots, y_{2n-1}\}$.

For each $i \in N_{<2n}$ and for each $j \in N_{<m+r}$, let $A_{i,j}$ be as in the proof of Proposition 4.3. For each $i \in N_{<2n}$, let $K_i = \langle S_i, \mho_i, \Im_i, \Delta_i, \iota_i \rangle$ be an LTS such that $S_i = \{0, 1, 2, 3\}$, $\iota_i = \{2\}$, and $\mho_i = \emptyset$. For each $i \in N_{<n}$, let $\Im_i = N_{<m} \cup \{-1\}$ and
$\Delta_i = (\cup_{j=0}^{m-1} \{2\} \times \{j\} \times A_{i,j}) \cup \{\langle 2, -1, 3 \rangle\}$. For each $i \in (N_{<2n}) \setminus (N_{<n})$, let
$\Im_i = ((N_{<m+r}) \setminus (N_{<m})) \cup \{-2\}$ and $\Delta_i = (\cup_{j=m}^{m+r-1} \{2\} \times \{j\} \times A_{i,j}) \cup \{\langle 2, -2, 3 \rangle\}$.

Now $\langle K_0, \ldots, K_{2n-1} \rangle$ is action-consistent. For the initial state in the parallel composition of $\langle K_0, \ldots, K_{2n-1} \rangle$, for each $j \in N_{<m}$, the number of successor transitions labelled by j is equal to the number of total assignments of $\{y_0, \ldots, y_{n-1}\}$ satisfying C_j, whereas for each $j \in (N_{<m+r}) \setminus (N_{<m})$, the number of successor transitions labelled by j is equal to the number of total assignments of $\{y_n, \ldots, y_{2n-1}\}$ satisfying C_j. The rest follows from the fact that $N_{<m} \cup \{-1\}$ and $((N_{<m+r}) \setminus (N_{<m})) \cup \{-2\}$ are CFFD-stubborn at the pair of the initial state and the empty set. □

Lemma 5.3 solves the main open problem of [21] by implying that as far as the complexity class P is not equal to NP, it is more difficult to minimize the number of successor states than the number of successor transitions. Lemma 5.3 also motivates the approach presented in [21] by indicating that it is very difficult or perhaps even impossible to avoid #P-complete counting in the general case.

Proposition 5.4 *CFFD2ELIG is PP-complete.*

Proof. Due to Lemma 5.3, it suffices to show that CFFD2ELIG is in PP. Let s, Q and R be as in the formulation of CFFD2ELIG. Let Y be the set of those successor states of s that are produced by Q but not by R. Respectively, let Z be the set of those successor states of s that are produced by R but not by Q. However, if the assumptions of CFFD2ELIG are not satisfied, we force $Y = \{s\}$ and $Z = \emptyset$. The assumptions of CFFD2ELIG can be checked in polynomial time, whereas for any state, membership checks w.r.t. Y and Z can be done in polynomial time.

We obtain a polynomial time probabilistic Turing machine that works as follows and is deterministic in every respect except coin tossing. We assume an encoding that is uniquely decodable in polynomial time and able to encode any state of the parallel composition into a bit vector of t bits where t depends on the input but is fixed for any fixed input. Moreover, we require that 2^t is greater than the number of the states. Vectors that are not images of states w.r.t. the encoding are images of appropriately defined "non-states". The machine starts by tossing a coin $2t + 2$ times. After that, the machine interprets the first t coin tosses as a vector of t bits and decodes the vector w.r.t. the encoding. If the result of the decoding is not in $Y \cup Z$, the machine halts in such a way that an outcome "tails" of the last coin toss causes acceptance, an outcome "all heads" of the last $t + 2$ coin tosses causes acceptance, and anything else causes rejection. If the result of the decoding is in Y, the machine halts in such a way that an outcome "all heads" of the last $t + 2$ coin tosses causes acceptance and anything else causes rejection. If the result of the decoding is in Z, the machine halts with acceptance.

It follows that for any input, the number of accepting computations is
$\mu = (2^{t+1} + 1)(2^t - |Y| - |Z|) + |Y| + 2^{t+2}|Z|$, whereas the number of other, i.e. rejecting, computations, is $\nu = 2^{2t+2} - \mu = (2^{t+1} - 1)(2^t - |Y| - |Z|) + (2^{t+2} - 1)|Y|$.
Now $\mu - \nu = 2(2^t - |Z|) + 2^{t+2}(|Z| - |Y|)$. If $|Y| \leq |Z|$, then $\mu - \nu \geq 2(2^t - |Z|) > 0$.
If $|Y| > |Z|$, then $\mu - \nu \leq 2(2^t - 2^{t+1} - |Z|) = -2(2^t + |Z|) < 0$. □

6 Conclusions

Due to the general nature of Proposition 4.3, it may be that some very close result has been published before. However, the author of the present paper has never encountered such a publication. Lemma 5.3 can be thought of as a "moral corollary" of Proposition 4.3, but the claim is certainly less than obvious and actually relies on the possibility to synchronize invisible actions. (Synchronization of invisible actions was allowed already in [18] and further motivated in [21].)

As suggested by the word "towards" in the title of this paper, we are really taking initial steps. The state-of-art in the known implementations of partial order reduction methods is that heuristics concentrate on optimizing the behaviour in "typical" cases. Proceeding into that direction leads to more and more application specific implementations. Proceeding into the "generic and ambitious" direction should improve tools that are to be used for "as many purposes as possible".

Acknowledgements

Many parts in this paper have benefited from ideas and references provided by Pekka Orponen. Proposition 4.3 resulted from observations made by Pekka Orponen, Antti Huima and the author.

References

[1] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag, Berlin, 1999. Related information is available via http://www.nada.kth.se/theory/compendium/wwwcompendium.html.

[2] Cook, S. A.: The Complexity of Theorem-Proving Procedures, in: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, ACM, New York NY, USA, 1971, 151–158.

[3] Gill, J.: Computational Complexity of Probabilistic Turing Machines, *SIAM Journal on Computing*, **6**(4), 1977, 675–695.

[4] Gusfield, D., Pitt, L.: A Bounded Approximation for the Minimum Cost 2-Sat Problem, *Algorithmica*, **8**, 1992, 103–117.

[5] Hochbaum, D. S., Megiddo, N., Naor, J., Tamir, A.: Tight Bounds and 2-Approximation Algorithms for Integer Programs with Two Variables Per Inequality, *Mathematical Programming*, **62**, 1993, 69–83.

[6] Hochbaum, D. S., Ed.: *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, Boston MA, USA, 1997.

[7] Jonsson, P.: Tight Lower Bounds on the Approximability of Some NPOPB-Complete Problems, *Linköping Electronic Articles in Computer and Information Science*, **2**(4), 1997, http://www.ep.liu.se/ea/cis/1997/004/.

[8] Kaivola, R.: *Equivalences, Preorders and Compositional Verification for Linear Time Temporal Logic and Concurrent Systems*, Doctoral Thesis, University of Helsinki, Department of Computer Science, Report A-1996-1, March 1996.

[9] Kann, V.: Polynomially Bounded Minimization Problems that are Hard to Approximate, *Nordic Journal of Computing*, **1**, 1994, 317–331.

[10] Karp, R. M., Luby, M., Madras, N.: Monte-Carlo Approximation Algorithms for Enumeration Problems, *Journal of Algorithms*, **10**(3), 1989, 429–448.

[11] Motwani, R., Raghavan, P.: *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[12] Orponen, P., Mannila, H.: *On Approximation Preserving Reductions: Complete Problems and Robust Measures*, University of Helsinki, Department of Computer Science, Report C-1987-28, 1987. There is also a revised version (May 1990): http://www.tcs.hut.fi/%7eorponen/papers/approx.ps.

[13] Papadimitriou, C. H.: *Complexity Theory*, Addison-Wesley, Reading MA, USA, 1994.

[14] Sahni, S.: Computationally Related Problems, *SIAM Journal on Computing*, **3**(4), 1974, 262–279.

[15] Simon, J.: *On Some Central Problems in Computational Complexity*, PhD Thesis, Cornell University, Department of Computer Science, Technical Report TR 75-224, Ithaca NY, USA, January 1975.

[16] Valiant, L. G.: The Complexity of Enumeration and Reliability Problems, *SIAM Journal on Computing*, **8**(3), 1979, 410–421.

[17] Valmari, A.: *Alleviating State Explosion during Verification of Behavioural Equivalence*, University of Helsinki, Department of Computer Science, Report A-1992-4, August 1992.

[18] Valmari, A.: Stubborn Set Methods for Process Algebras, in: *Partial Order Methods in Verification* (D. A. Peled, V. R. Pratt, G. J. Holzmann, Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 29, American Mathematical Society, Providence RI, USA, 1997, 213–231.

[19] Valmari, A., Tienari, M.: Compositional Failure-Based Semantic Models for Basic LOTOS, *Formal Aspects of Computing*, **7**(4), 1995, 440–468.

[20] Varpaaniemi, K.: Stable Models for Stubborn Sets, *Fundamenta Informaticae*, **43**(1–4), 2000, 355–375.

[21] Varpaaniemi, K.: Minimizing the Number of Successor States in the Stubborn Set Method, *Fundamenta Informaticae*, **51**(1–2), 2002, 215–234.