REMARKS ON THE SATISFIABILITY PROBLEM OF PROPOSITIONAL LOGIC *

by

Hans-Georg Stork[X)]

* WG 78 (Graphs - Data Structures - Algorithms), Hanser 1979

X) Institut für Informatik der Univ. Stuttgart, Azenbergstrasse 12, D-7000 Stuttgart 1

Abstract: The problem of finding classes of "simple" inputs for the Davis-Putnam algorithm is explained and illustrated. An infinite class of inputs is called "simple" with respect to DPA if the runtime of DPA on each element of that class is bounded from above by some polynomial function of the size of that element.

The satisfiability problem of propositional logic has been the first combinatorial problem to be shown NP-complete ([1]). It can be stated as follows:

Given a potentially infinite set $X = \{x_1, x_2, \ldots\}$ of propositional variables. A clause is a finite set

$$C = \{y_1^{\varepsilon_1}, \ldots, y_k^{\varepsilon_k}\},$$

where: Cl-1: $y_i \in X$, $(i = 1,\ldots,k)$, $y_i \neq y_j$ for $i \neq j$

Cl-2: $\varepsilon_i \in \{+,-\}$, $(i = 1,..,k)$

Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ $(m \geq 1)$ be a set of clauses (a SOC), let $X_{\mathcal{C}} \subset X$ be the set of variables occurring in the C_i $(i = 1,\ldots,k)$, and let $v: X \to \{0,1\}$ be an assignment of boolean values. Then v is extended to an assignment $v(\mathcal{C}) \in \{0,1\}$ as follows:

A-1: $v(x) = 0 \Rightarrow v(x^+) = 0$ and $v(x^-) = 1$

$v(x) = 1 \Rightarrow v(x^+) = 1$ and $v(x^-) = 0$, for $x \in X$

A-2:
$$v(C) = \begin{cases} 1 & \text{if } v(y^\varepsilon) = 1 \text{ for some } y^\varepsilon \in C \\ 0 & \text{otherwise} \end{cases}$$

A-3:
$$v(\mathcal{C}) = \begin{cases} 1 & \text{if } v(C) = 1 \text{ for all } C \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

In case $v(\mathcal{C}) = 1$ we say $\mathcal{C}$ is satisfied by the assignment v.

The satisfiability problem (SP) is to decide, given a SOC $\mathcal{C}$, whether or not there exists an assignment v that satisfies $\mathcal{C}$.

The justification for including a talk on this problem in a meeting of graph-theorists is twofold: First, it is well known ([5]) that the satisfiability problem can be reduced in deterministic polynomial time to a number of interesting graph-theoretical problems, like clique-finding, graph-colouring etc. Secondly, there is an immediate graph interpretation of the problem itself:

Let $G = (V,E)$ be a bipartite graph, where:

G-1: $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$ is the set of vertices

G-2: $E = E_+ \cup E_- \subset \{\{a,b\} \mid a,b \text{ not both in } V_i,\ i = 1,2\}$
$E_+ \cap E_- = \emptyset$, is the set of edges

G-3: For each $b \in V_2$ there is at most one $a \in V_1$ such that $\{a,b\} \in E$

G-4: There are no isolated vertices

For $\varepsilon \in \{+,-\}$ and $a \in V_1$ let $E_\varepsilon(a) = \{\{a,b\} \mid b \in V_2, \{a,b\} \in E_\varepsilon\}$

Now find a subset $E' \subset E$ of edges with the following properties:

S-1: $E' = \bigcup_{a \in V_1} E_{\varepsilon_a}(a)$, for some selection $\varepsilon_a \in \{+,-\}$

S-2: The graph $G' = (V,E')$ contains no isolated vertices

It is quite obvious that the problem to decide, given $G = (V,E)$ with properties G-1 - G-4, whether or not there exists $E' \subset E$ with S-1 and S-2, is equivalent to the aforementioned satisfiability problem.

For a SOC $\mathcal{C} = \{C_1, \ldots C_m\}$ we define the length of $\mathcal{C}$ to be

$l(\mathcal{C}) = \sum_{i=1}^{m} |C_i|$, where $|C_i|$ is the cardinality of C_i.

SP being NP-complete, it is not known whether or not there exists a deterministic algorithm A whose runtime T_A (on some ideal computer <u>without</u> unbounded inherent parallelism), when applied to some SOC $\mathcal{C}$, is bounded from above by the value of some polynomial function of $l(\mathcal{C})$. However, the following could be shown ([3]) for a certain class $\mathcal{A}_0$ of algorithms solving SP:

<u>Theorem</u>: ([3],[7]) There exists an infinite set $\mathcal{L}_0$ of SOC's and constants $c,d \geq 0$ such that for all $A \in \mathcal{A}_0$ and for all $\mathcal{C} \in \mathcal{L}_0$:

$T_A(\mathcal{C}) \geq c2^{d \cdot l(\mathcal{C})}$.

The class $\mathcal{A}_0$ is based on what has become known as "regular resolution procedures" ([3],[6],[7]).
$\mathcal{A}_0$ happens to include a well-known algorithm due to Davis and Putnam ([2]) for solving SP (from now on abbreviated DPA). Since in what follows we are only interested in this latter algorithm, $\mathcal{A}_0$ will not be explicated further. For reference see [3].
Having found an infinite set of SOC's that are "complex" with respect to DPA (i.e. on which DPA requires exponential time) the natural question arises whether nontrivial infinite sets of SOC's can be distinguished that are "simple" with respect to DPA (i.e. which

can be "solved" by DPA in polynomial time). This type of question has come up in other branches of computer science as well. For instance: How do programs have to be constructed in order to be amenable to uniform procedures for checking their correctness? A question that is underlying the discipline of "Structured Programming". Or: What should the grammar of a programming language look like in order to allow a "fast" syntactical analyzer for programms written in that language? A question that has stimulated considerable research on special types of context-free grammars, like LR(k) etc. It should be noted however that the question we are posing has a somewhat different flavor: we are given one particular algorithm with respect to which we are looking for classes of "simple" inputs. In contrast to that, the search for well-structured programs aims at distinguishing a class of programs for which a problem (namely checking correctness) is algorithmically solvable, whereas the general problem in this case is not effectively solvable at all. Secondly, "good" grammars are usually not defined with respect to syntactical analyzers that would work for general context-free grammars as well; they rather demand their own specialized analyzer.

Before indicating two known nontrivial sets of "simple" SOC's (one of which is almost trivial) and a procedure for constructing a third one, we shall formulate SP in terms of a special class of matrices over {+,-,0}(a formulation that has been suggested in [4], and used therein for the representation of SOC's in computer memories).

Let $\mathcal{M}$ = {M| M is an n×m-matrix over {+,-,0}; n,m $\geq$ 1; no row and no column of M consists entirely of 0's} $\cup$ {(∅)}

M ε $\mathcal{M}$ will sometimes be denoted as:

$M = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix}$, where $r_i = (a_{i1},\ldots,a_{in})$ is row-vector over {+,-,0},

or as $M = (c_1,\ldots,c_n)$, where $c_i = \begin{pmatrix} a_{1i} \\ \vdots \\ a_{mi} \end{pmatrix}$ is a column over {+,-,0};

(∅) is the "empty" matrix, containing no rows and no columns.

Definition: Let M ε $\mathcal{M}$. M is called satisfiable iff either M = (∅) or there exists a row-vector $r_o = (a_{o1},\ldots,a_{on})$ of dimension n over {+,-} such that: for all i ε {1,...,m} there exists j ε {1,...,n} with $a_{oj} = a_{ij}$.

The satisfiability-value $\sigma(M)$ of $M \in \mathcal{M}$ is defined as

$$\sigma(M) = \begin{cases} 1 & \text{if } M \text{ is satisfiable} \\ 0 & \text{otherwise} \end{cases}$$

Obviously, if $M \in \mathcal{M}$ is an $n\times 1$-matrix then $\sigma(M) = 1$. On the other hand, if $M \in \mathcal{M}$ is a $1\times m$-matrix, $M = (c_1)$, then

$$\sigma(M) = \begin{cases} 1 & \text{if } c_1 \text{ is either over } \{+\} \text{ or over } \{-\} \\ 0 & \text{otherwise} \end{cases}$$

The following observation is trivial:
Remark: Let $\mathcal{C} = (C_1,\ldots,C_m)$ be a SOC with $|V_{\mathcal{C}}| = n$. Then there is an $n\times m$-matrix $M \in \mathcal{M}$ such that $\sigma(M) = 1$ iff $\mathcal{C}$ is satisfiable, and vice versa.

Instead of giving the details of the translations we describe them by an easily generalizable example:
Example: Let $\mathcal{C}$ be given by $C_1 = \{x_1^+, x_2^-, x_4^+\}$, $C_2 = \{x_1^-, x_2^-, x_3^+\}$, $C_3 = \{x_2^+, x_4^-\}$. Then

$$M = \begin{bmatrix} + & - & 0 & + \\ - & - & + & 0 \\ 0 & + & 0 & - \end{bmatrix}, \text{ and vice versa.}$$

Here we have $\sigma(M) = 1$, since $r_o = (+,+,+,+)$ satisfies M.

If $M \in \mathcal{M}$ is an $n\ m$-matrix we define its size by $s(M) = nm$.
If $r_i = (a_{i1},\ldots,a_{in})$ is a row of M then its length $l(r_i)$ is the number of its non-zero elements.
Clearly, if there is a polynomial-time (in $s(M)$) algorithm for deciding the satisfiability-value of $M \in \mathcal{M}$, then there is a polynomial-time (in $l(\mathcal{C})$) algorithm for solving SP, and vice versa.

For two rows $r_i = (a_{i1},\ldots,a_{in})$ and $r_j = (a_{j1},\ldots,a_{jn})$ of $M \in \mathcal{M}$ we say "r_i is contained in r_j" iff $a_{ik} \neq 0 \Rightarrow a_{ik} = a_{jk}$, for $k = 1,..,n$
In order to formulate DPA in terms of $\mathcal{M}$ we still have to define an operation "$\bar{v}$" on $\{+,-,0\}$ and to extend it to row-vectors:
$\bar{v}$ is defined by the following table, where "u" means undefined:

$\bar{v}$	+	-	0
+	+	u	+
-	u	-	-
0	+	-	0

Let r_i, r_j be as above (i.e. with equal dimension):

$$r_i \bar{v} r_j = \begin{cases} (a_{i1} \bar{v} a_{j1},\ldots,a_{in} \bar{v} a_{jn}) & \text{if all components are defined} \\ \text{undefined, otherwise} \end{cases}$$

Davis-Putnam Algorithm on $\mathcal{M}([2])$:

Let $M \in \mathcal{M}$, $M = \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix} = (c_1,\ldots,c_n)$, $c_j = \begin{bmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{bmatrix}$, $r_i=(a_{i1},\ldots,a_{in})$ be given.

step 1: IF $M = (\emptyset)$ or $m = 1$ THEN $\sigma(M) = 1$; STOP
ELSE proceed
IF there are r_i, r_j such that $0 \neq a_{ik} \neq a_{jk} \neq 0$ for some $k \in \{1,\ldots,n\}$ and $a_{il} = a_{jl} = 0$ for $l \neq k$
THEN $\sigma(M) = 0$; STOP
ELSE proceed

step 2: ("subsumption rule") FOR all r_i, r_j, $i \neq j$ DO
IF r_i is contained in r_j THEN delete r_j
("If M' is the matrix obtained from M by step 2 then $\sigma(M') = \sigma(M)$")

step 3: IF there is r_i such that $a_{ik} \neq 0$ for some $k \in \{1,\ldots,n\}$ and $a_{il} = 0$ for $l \neq k$
THEN delete all rows r_j where $a_{jk} = a_{ik}$; delete column c_k; GOTO step 1
ELSE proceed
("For M' obtained from M by step 3 $\sigma(M') = \sigma(M)$")

step 4: IF there is c_k such that for $i,j \in \{1,\ldots,m\}$ $a_{ik} \neq 0$ and $a_{jk} \neq 0 \Rightarrow a_{ik} = a_{jk}$
THEN delete all r_i where $a_{ik} \neq 0$; delete c_k; GOTO step 1
ELSE proceed
("As above $\sigma(M') = \sigma(M)$")

step 5: select $k \in \{1,\ldots,n\}$, let $R_\varepsilon = \{r_i \mid a_{ik} = \varepsilon\}$, $\varepsilon \in \{+,-\}$,
for $r_i = (a_{i1},\ldots,a_{in})$ let $\bar{r}_i = (a_{i1},\ldots,a_{i,k-1},a_{i,k+1},\ldots,a_{in})$
delete all $r_i \in R_+ \cup R_-$; delete c_k;
FOR all $r_i \in R_+$ and $r_j \in R_-$ DO insert $\bar{r}_i \vee \bar{r}_j$ if defined;
GOTO step 1
("As above $\sigma(M') = \sigma(M)$")

Remark: Originally, DPA has been stated without the subsumption rule of step 2; one of its effects will be seen later.

Suppose now we are given an "ideal computer without unbounded inherent parallelism", and let $T_{DP}(\cdot)$ be the runtime of DPA on this machine.

Definition: An infinite subset $\mathcal{M}_0 \subset \mathcal{M}$ is called DP-simple iff there

exists a polynomial function $p = p(n)$, $n \in \mathbb{N}$, such that for all $M \in \mathcal{M}_0$: $T_{DP}(M) \leq p(s(M))$.

Note that the class of DP-simple sets is closed with respect to intersection and finite union.

Before identifying a first DP-simple set we observe that only step 5 of DPA can be responsible for taking an infinite subset of $\mathcal{M}$ out of the realm of polynomiality: although each of the steps 1-5 can be performed in polynomial time with respect to the size of the current matrix, it may happen - in the worst case - that step 5 outputs a matrix of considerably increased size (up to $(n-1)m^2/4$) for further processing. On the contrary, steps 1-4 each reduce the size of the matrix at hand. Thus, if in $\mathcal{M}' \subset \mathcal{M}$ there occurs every once in a while a matrix which requires too many executions of step 5 $\mathcal{M}'$ is likely not to be DP-simple.

The following fact is well known:
Proposition: The set $\mathcal{M}_0 = \{M \mid M \in \mathcal{M}, l(r) \leq 2$ for all rows r of $M\}$ is DP-simple.

In fact, one iteration of DPA on any $n \times m$-matrix $M \in \mathcal{M}_0$ yields a matrix $M' \in \mathcal{M}_0$ whose size is at most $(n-1)m'$, where m' is the maximal number of different rows r with $l(r) \leq 2$ and dimension $n-1$ that may be contained in a matrix with $n-1$ columns. m' is quadratic in $n-1$.

Evidently, the same argument cannot be applied to sets $\mathcal{M}' \subset \mathcal{M}$ whose elements do not satisfy the above mentioned length condition on its rows: if there are rows r_i, r_j with $l(r_i) = l(r_j) = 3$ then step 5 may produce a row of length 4.

The second DP-simple set to be exhibited here is due to Harris([4]) Let $r = (a_1, \ldots, a_n)$ be a row over $\{+,-,0\}$ of dimension n, let $e,f \in \mathbb{N}$ such that $e+f \leq n$. Then r is said to be of type (e,f) iff r contains exactly e +'s and f -'s.
Proposition ([4]): The set $\mathcal{M}_0 = \{M \mid M \in \mathcal{M}$, if M contains a row of type (e,f) then M contains all rows of type (e,f)$\}$ is DP-simple.

The proof of this proposition is essentially along the same line as the proof of the aforementioned result: first it can be shown that one iteration of DPA on an $n \times m$-matrix $M \in \mathcal{M}_0$ again yields an an element $M' \in \mathcal{M}_0$(i.e. $\mathcal{M}_0$ is invariant with respect to iterations of DPA). Secondly, by a careful inspection of the row-types present

in the new matrix, a $O(nm^2)$ bound can be found for the size of any matrix occurring after DPA has been started on M. This together with the fact that DPA iterates at most n times yields the polynomial bound of DPA as applied to $\mathcal{M}_0$.

It seems that these two propositions, or rather their proofs, indicate a general method for obtaining DP-simple sets of matrices. However, to my knowledge, no further examples of this kind have appeared in the literature.
Another approach might be feasible: Try to explicitly generate DP-simple sets $\mathcal{M}_0 \subset \mathcal{M}$ by reversing the DPA-rules in a restricted manner. Then the following problems may be posed:

1. Characterize $\mathcal{M}_0$ in "static" terms, i.e. not in terms of the generation procedure.
2. Count or estimate how many n×m-matrices in $\mathcal{M}$ are also contained in $\mathcal{M}_0$, in order to obtain a "measure of density" of $\mathcal{M}_0$ with respect to $\mathcal{M}$.

Recalling the observation following the definition of DP-simplicity we define:

Definition: A matrix $M \in \mathcal{M}$ is called solvable by reduction (RED-solvable) iff DPA applied to M uses its steps 1-4 only.

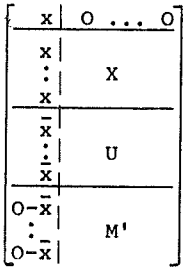
Let $\mathcal{M}_{RED} = \{M \mid M \in \mathcal{M}, M \text{ is RED-solvable}\}$. Obviously $\mathcal{M}_{RED}$ is a DP-simple set.

In the remainder we present an algorithm that, for given $n,m \in \mathbb{N}$, constructs all n×m-matrices in $\mathcal{M}_{RED}$. It works by essentially applying steps 1-4 of DPA in reverse.

In case some column is deleted by either step 3 or step 4 we say that this column has been 3-eliminated (resp. 4-eliminated). The reverse of course is to create new columns that can be either 3-eliminated or 4-eliminated. Correspondingly, we speak of 3-creation and 4-creation. Since any elimination is preceded by subsumption 3-creation (resp. 4-creation) applied to some matrix M' will be such that applying subsumption and then eliminating the newly created column yields a submatrix of M'. Also, by reversing step 1, any matrix can be immediately "falsified". It may happen though, that DPA uses its step 5 even if started on a matrix that has been constructed by consecutively creating new columns. We have to show that - due to the very formulation of DPA - this is in fact not the

case. First let us explicitly define the i-creations (i = 3,4):

3-creation: Let M' be an $(n-1)\times m_0$-matrix. Then any matrix M that can be obtained from



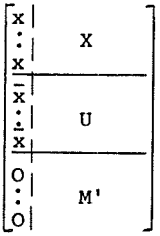
(where: $x \in \{+,-\}$, $\bar{x} = \begin{cases} + & \text{if } x = - \\ - & \text{if } x = + \end{cases}$,

X is an arbitrary $(n-1)\times m_1$-matrix,

U is an $(n-1)\times m_2$-matrix whose rows contain rows of M'; $0\text{-}\bar{x}$ means: either 0 or $\bar{x}$ can be inserted)

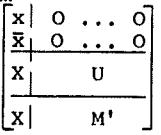
by applying arbitrary permutations to its rows and to its columns, is called 3-created from M'.

4-creation: Let M', $x, \bar{x}$, X and U be as above. Then any matrix M that can be obtained from



by applying arbitrary permutations to its rows and to its columns, is called 4-created from M'.

Falsification: Let M' etc. be as above. Then any matrix M obtained from



by arbitrary permutations of rows and columns is called a falsification of M'.

M' may be the empty matrix; in this case U is also empty whenever it occurs.

Denote the operations 3-creation, 4-creation and falsification by C3, C4, and C1 respectively, the corresponding eliminations by E3, E4, and E1.

For cr $\in$ {C1,C3,C4} we write M = cr(M') if M has been constructed from M' by the operation cr. (Actually, cr should be considered a "nondeterministic operator" since its application may yield a multitude of outcomes.)

The above claim is now summarized in the following lemma:

Lemma: Let $M_o \in \mathcal{M}$ and let $M_1 = cr(M_o)$, where $cr \in \{C1,C3,C4\}$. Let DPA during its first iteration eliminate by some el $\in\{E1,E3,E4\}$ a column $j \neq k$, where k is the number of the column in M_1 created by cr. Then: Either (i) $\sigma(M_1) = O$

or (ii) a matrix M_1' is obtained such that

either (ii1) $M_1' = cr(M_o')$ for some other matrix M_o'

or (ii2) in M_1' column k has been eliminated as well.

Proof: Without loss of generality, $k = 1$. Hence M_1 is of one of the three forms indicated in the definitions of C3, C4 and C1 respectively.

(1) If el = E1 then $\sigma(M_1) = O$, proving (i).

(2) If el $\neq$ E1 then cr $\neq$ C1, since otherwise DPA would have eliminated the first column, a contradiction. Hence $cr \in \{C3,C4\}$.

(a) Let cr = C3; then el = E3, since by DPA an elimination E3 is performed before E4 is performed. Hence we have the following situation:

$$M_1 = \begin{array}{c|c|c|c} x & O\ldots & O & \ldots O \\ \hline x & & & \\ \vdots & X & & \\ x & & & \\ \hline \bar{x} & & & \\ \vdots & U & & \\ \bar{x} & & & \\ \hline & M_o & & \\ O & O\ldots & y & \ldots O \end{array}$$

1 j

Since elimination of column j does not affect the first row of M_1 (ii) is proved in this case.

(b) If cr = C4 then, by inspection of DPA, either el = E3 or el = E4.

(b1) Let el = E3. M_1 must be of the form

$$M_1 = \begin{array}{|c|c|c|} x & X & \\ \vdots & & \\ x & & \\ \hline \bar{x} & & \\ \vdots & U & \\ \bar{x} & & \\ \hline O & O\ldots & y \ldots O \\ \vdots & M_o & \\ O & & \end{array}$$

1 j

Applying subsumption first, we get:

$$\bar{M}_1 = \begin{array}{|c|c|c|} x & \bar{X} & \\ \vdots & & \\ x & & \\ \hline O & O\ldots & y \ldots O \\ \vdots & \bar{M}_o & \\ O & & \end{array}$$

1 j

and this structure is not destroyed, by eliminating column j, provided the rows corresponding to $\bar{X}$ have not been deleted; hence (ii2) may happen in this case. This proves (ii) for b1.

(b2) Let el = E4. Then M_1 must be of the form

$$M_1 = \begin{array}{c|c|c} x & X & y \\ \vdots & & \vdots \\ x & & y \\ \hline \bar{x} & U & \bar{y} \\ \vdots & & \vdots \\ \bar{x} & & y \\ \hline 0 & & \\ \vdots & M_o & \bar{y} \\ 0 & & \end{array}$$

(columns: 1, j)

with each row whose j-th element is $\bar{y}$ being a superrow of some other row containing a 0 in its j-th column. Subsumption certainly deletes the rows corresponding to U, and since E4 is applicable to $j \neq k$, step 2 of DPA yields:

$$\bar{M}_1 = \begin{array}{c|c|c} x & \bar{X} & \\ \vdots & & y \\ x & & y \\ \hline 0 & & \\ \vdots & \bar{M}_o & y \\ 0 & & y \end{array}$$

(columns: 1, j)

with column j containing only either +'s and 0's or -'s and 0's. Hence deleting column j and the rows corresponding to its nonzero elements does not destroy the structure of $\bar{M}_1$ if not at the same time the rows corresponding to $\bar{X}$ are deleted, whence case (ii2) may occur in this situation as well.

Remark: Note that for the above lemma the very formulation of DPA is essential: if the subsumption-step were performed after 3-elimination it would no longer be guaranteed that a 4-creation can be reversed by a corresponding 4-elimination. Hence the "timing" of the subsumption-step within DPA considerably (?) affects the efficiency of this algorithm.

The lemma now allows to characterize the DP-simple set $\mathcal{M}_{RED}$ in terms of the operations C1, C3 and C4:

Proposition: The following statements are equivalent:

(i) $M \in \mathcal{M}$ is RED-solvable

(ii) $M = M_o$ or $M = cr_n \circ cr_{n-1} \circ \ldots \circ cr_1(M_o)$, $n \geq 1$
for some M_o that is either empty or a row or a column.

("$\circ$" denotes subsequent application; $cr_i \in \{C1, C3, C4\}$, $i = 1,..,n$)

Proof: (i) $\Rightarrow$ (ii) is true by definition . (ii) $\Rightarrow$ (i) follows from the lemma: any elimination performed by DPA other than in the order given by the creations either renders other eliminations superfluous ((ii2) above) or does not affect their later applicability ((ii1) above).

The latter proposition immediately yields an algorithm for constructing any RED-solvable n×m-matrix in $\mathcal{M}$:

Given: $n,m \in \mathbb{N}$

step 1: generate a strictly increasing sequence of natural numbers
$m_1 < m_2 < \ldots < m_k = m$, such that
(i) $k \leq n$ (ii) $k \neq n \Rightarrow m_1 = 1$

step 2: in case $k \neq n$ let M_1 be a row of dimension $n-k+1$,
in case $k = n$ let M_1 be a column of dimension m_1 ($M_1 \in \mathcal{M}$)

step 3: FOR $i = 2,\ldots,k$ DO
IF M_{i-1} is an $n_{i-1} \times m_{i-1}$-matrix THEN
IF $m_i - m_{i-1} > 1$ let $M_i = cr(M_{i-1})$ be some $(n_{i-1}+1) \times m_i$-matrix, where $cr \in \{C1,C3,C4\}$
ELSE let $M_i = cr(M_{i-1})$ be some $(n_{i-1}+1) \times m_i$-matrix where $cr \in \{C3,C4\}$

Remark: Note that $n_{k-1} + 1 = n$. In step 3 the two cases have to be distinguished since C1 is applicable only if $m_i - m_{i-1} \geq 2$.

Conclusion: We illustrated the problem of finding classes of simple inputs for a given algorithm A whose worst-case runtime is known to be exponential. For SP and DPA we have indicated two such classes in "static" terms and exhibited a procedure for constructing a third one. Among others, the following problems might also be of interest in connection with A-simple sets:

1. Let $\mathcal{M}_0$ be A-simple and A' some other algorithm for solving SP. What is the behaviour of $\mathcal{M}_0$ with respect to A'?
2. For SP a natural optimization problem is to compute the maximal number of clauses that can be satisfied (and actually construct a maximal set). Given an algorithm A' for doing this approximately and some A-simple $\mathcal{M}_0$. How does A' perform on $\mathcal{M}_0$?
3. Let $\mathcal{P}$ be some NP-problem to which SP can be reduced by some polynomial algorithm A', let $\mathcal{M}_0$ be A-simple. How can $A'(\mathcal{M}_0)$ be characterized in terms of $\mathcal{P}$ or in terms of an algorithm for solving $\mathcal{P}$ that does not implicitly use the reduction to SP?
4. If $\mathcal{M}_0$ is A-simple its "density" with respect to $\mathcal{M}$ may be defined as follows:
$$d(\mathcal{M}_0) = \limsup_{s \to \infty} \frac{|\{M \mid M \in \mathcal{M}_0, \sigma(M) = s\}|}{|\{M \mid M \in \mathcal{M}, \sigma(M) = s\}|}$$
Then, if not for all A and A-simple $\mathcal{M}_0$ $d(\mathcal{M}_0) = 0$, the quality of algorithms (for SP, but mutatis mutandis for other NP-problems as well) may be judged by the density of their simple sets.

References:

[1] Cook S.A.: The complexity of theorem proving procedures; 3rd ACM Symp. on Theory of Computing; 1971

[2] Davis M.,H.Putnam: A computing procedure for quantification theory; Journal ACM Vol.7; 1960

[3] Galil Z.: The complexity of resolution procedures for theorem proving in the propositional calculus; Ph.D.thesis, Cornell University; 1975

[4] Harris R.V.: A polynomial bound on the complexity of the Davis-Putnam algorithm applied to symmetrizable propositions; Ph.D. thesis, Cornell University; 1972

[5] Karp R.M.: Reducibility among combinatorial problems; in: Complexity of Computer Computations, R.E. Miller and J.W. Thatcher (eds.); Plenum Press, New York; 1972

[6] Robinson J.A.: A machine oriented logic based on the resolution principle; Journal ACM Vol.12; 1965

[7] Tseitin G.S.: On the complexity of derivations in the propositional calculus; in: Structures in constructive mathematics and mathematical logic, part II, A.O. Silenko (ed.); 1968(Translated from Russian)