4. The efficiency of SSS* relative to α-β

In this section it is shown that the SSS* algorithm dominates the α-β algorithm in the sense that whenever SSS* must explore a node then so must α-β. It is also shown for practical distributions of tip values that SSS* is strictly superior to α-β in terms of average number of tip nodes examined. Prior to proving these results interesting examples are presented which motivated the search for those results. In addition, simulations performed before deriving the theorems are briefly discussed.

4.1 Interesting examples comparing α-β and SSS*

Very early in the study of SSS* example game trees were produced in which SSS* explored fewer nodes than α-β. An example particularly unfavorable to α-β is given in Figures 1 and 3: SSS* ignores 11 nodes while α-β ignores only 3. α-β does not do well because the best path is toward the right of the tree.

Table II lists all possible (ranked) distinct tip assignments to a 2-level binary game tree and the number of tips that α-β and SSS* would explore. The average number of tips explored is 11/3 for α-β and 10/3 for SSS*. More important is the fact that SSS* never explores a node that α-β ignores.
Table II  SSS* domination of α-β in a 2-level binary game tree.

<table>
<thead>
<tr>
<th>Case</th>
<th>tip assignments</th>
<th>number of tips α-β evaluates</th>
<th>number of tips SSS* evaluates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1 2 4 3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1 3 2 4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1 3 4 2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1 4 2 3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1 4 3 2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>2 1 3 4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2 1 4 3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>2 3 1 4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>2 3 4 1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>2 4 1 3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>2 4 3 1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>3 1 2 4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>3 1 4 2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>3 2 1 4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>3 2 4 1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>3 4 1 2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>3 4 2 1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>4 1 2 3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>4 1 3 2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>4 2 1 3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>4 2 3 1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td>4 3 1 2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>4 3 2 1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

AVG.=11/3  AVG.=10/3
4.2 Simulation of $\alpha-\beta$ versus SSS$^*$

Examples such as those given above were at first studied with respect to an older and weaker version of SSS$^*$ (SSS). Trees were found on which $\alpha-\beta$ explored fewer nodes than the older SSS Algorithm. Simulation was done in order to test the hypothesis that on the average $\alpha-\beta$ explored more terminal nodes. Not only did the simulation work support that hypothesis but it also led to the development of SSS$^*$ which can be proven to dominate $\alpha-\beta$. The final simulation results appear in Table III. 1000 identical sets of distinct tip assignments were presented to $\alpha-\beta$ and SSS$^*$ and the behavior of the algorithms was recorded. Not only was there a smaller mean number of tips evaluated for SSS$^*$ but there was also a smaller standard deviation. The simulation program was then altered to check if $\alpha-\beta$ did better on any single set of tip assignments. 100 sets of tip assignments were checked for four different game trees as indicated in Table III. None of the cases checked showed SSS$^*$ exploring a node that $\alpha-\beta$ could ignore.

4.3 Domination of $\alpha-\beta$ by SSS$^*$

It will now be shown that SSS$^*$ is more efficient than $\alpha-\beta$ in terms of the average number of nodes explored. In order to do the comparison a mathematical characterization of $\alpha-\beta$ is required. The characterization of $\alpha-\beta$ used here is due to Baudet and represents the maturation of a long path [6,2,3,4] of research.
Table III Comparative performance of α-β and SSS* on 1000 sets of distinct tip values.

<table>
<thead>
<tr>
<th>Branching factor N</th>
<th># of AND node levels past the root K</th>
<th># Tips in sol. tree</th>
<th># Sol. trees</th>
<th>Published Comparisons</th>
<th>α-β Performance</th>
<th>SSS* Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>α-β Exact value from Newborn [3] †† or Baudet[1]†</td>
<td>α-β Lower bound from Knuth &amp; Moore [2]*</td>
<td>Avg. # of tips eval. and standard deviation</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3.67††</td>
<td>3.41</td>
<td>3.65</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>7.44††</td>
<td>6.42</td>
<td>7.41</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>17.67†</td>
<td>13.7</td>
<td>17.6</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1</td>
<td>10</td>
<td>37.4</td>
<td>56.1</td>
<td>35.6</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1</td>
<td>20</td>
<td>177††</td>
<td>102</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>12.04†</td>
<td>12.1</td>
<td>5.67</td>
</tr>
<tr>
<td></td>
<td>&amp; 3</td>
<td>2</td>
<td>9</td>
<td>45.20†</td>
<td>45.0</td>
<td>22.1</td>
</tr>
<tr>
<td></td>
<td>&amp; 5</td>
<td>2</td>
<td>25</td>
<td>227.35†</td>
<td>188</td>
<td>227</td>
</tr>
<tr>
<td></td>
<td>&amp; 2</td>
<td>3</td>
<td>8</td>
<td>37.16†</td>
<td>37.3</td>
<td>22.1</td>
</tr>
<tr>
<td></td>
<td>&amp; 3</td>
<td>3</td>
<td>27</td>
<td>248.56†</td>
<td>265</td>
<td>248</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>16</td>
<td>2</td>
<td>135</td>
<td>110</td>
<td>78.5</td>
</tr>
</tbody>
</table>

* lower bound for α-β without deep cutoffs
& SSS* dominated α-β for first 100 tip assignments
4.3.1 Definition of a game tree

The following definition of a game tree is used throughout this section. A game tree is an AND/OR tree whose root node is of type AND, all immediate successors of AND nodes are of type OR, and all immediate successors of OR nodes are of type AND. While these restrictions are unnecessary for SSS* they are conventional assumptions for minimax and α-β.

4.3.2 Encoding the nodes of the game tree

Dewey decimal notation can be used to represent game tree nodes. Let the root be encoded as the sequence 1. Then if sequence J represents any nonterminal node of the tree, J.j ;j=1,2,...,n represent the n immediate successors (sons) of J. Figure 3 gives an example of Dewey decimal encoding of nodes.

4.3.3 Negamax evaluation of nodes of the game tree

Negamax evaluation of nodes is made with respect to value to the player making the move rather than with respect to the player moving from the root position as with pure minimax. If node J has n sons then

\[ v(J) \equiv \max \{-v(J.i) : 1 \leq i \leq n\}. \]

If we assume that the root is an AND node then \( v(J) = g(J) \) for AND nodes (player at J is player at root) and \( v(J) = -g(J) \) for OR nodes (player at J is opponent of player at root) where \( g \) is the minimax value defined in Section 2.
4.3.4 Legacy of elder brothers of a node

For any node \( J.j \) of the game tree

\[
c(J.j) \equiv \max \{ -v(J.i) | 1 \leq i < j \}.
\]

Since \( J.i, 1 \leq i < j \) are the older brothers of \( J.j \), \( c(J.j) \) accounts for information provided to a node from the older brothers. If \( j=1 \) the set of elder brothers is empty and by convention \( c(J.1) = -\infty \). See Figure 3 for examples.

4.3.5 Static functions \( \alpha \) and \( \beta \).

Finally we define two values associated with a node \( J \) by the \( \alpha-\beta \) procedure.

\[
\alpha(J) \equiv \max \{ c(a) \mid a=J \text{ or } a \text{ is an ancestor of } J \text{ of the same type as } J \}
\]

\[
\beta(J) \equiv -\max \{ c(a) \mid a \text{ is an ancestor of } J \text{ of the opposite type from } J \}
\]

\( \alpha(1) = -\infty \) by definition, and \( \beta(1) = +\infty \) by convention. Examples are given in Figure 3.

\[1\]

Baudet has shown that the negamax \( \alpha-\beta \) algorithm of Knuth and Moore [3] will explore node \( J \) of a game tree if and only if \( \alpha(J) < \beta(J) \). Similar results
Figure 3. Game tree on which SSS* explores fewer nodes than α-β.

(Nodes not explored by α-β are 1.1.1.2, 1.2.1.1.2, 1.2.2.1.2.)

(Nodes not explored by SSS* are 1.1.1.1.2, 1.1.1.2.2, 1.1.2.1.2, 1.1.2.1.1.2, 1.1.2.1.2.2, 1.1.2.1.2.1, 1.2.1.1.2, 1.1.2.1.2, 1.1.2.1.2, 1.2.1.1.2.)
were also obtained by Fuller et al [2]. It is important to note that $\alpha(J)$ and $\beta(J)$ are defined for all nodes of the tree but will not in fact be computed by $\alpha-\beta$ for all nodes of the tree.

4.3.6 Relationship of $\hat{h}$ of SSS* to $\alpha(J)$ and $\beta(J)$.

Two lemmas are now established which characterize the relationship between the merit $\hat{h}$ of state $(J,L,\hat{h})$ when SSS* visits AND node $J$ and the static values $\alpha(J)$ and $\beta(J)$.

Lemma 1. For all AND nodes $J$ explored by SSS* $\alpha(J) = -\hat{h}(J)$.

Proof: $\hat{h}(J)$ is defined since SSS* visits node $J$. Recall that $\hat{h}(J)$ is the merit assigned when $J$ is first visited by SSS* with state $(J,LIVE,\hat{h})$. By definition $\alpha(J) \equiv \max \{ c(a) \mid a = J$ or $a$ is an AND ancestor of $J \}$.

$= \max \{ \max \{ -v(e) \mid e$ is an elder brother of $a \} \mid a = J$ or $a$ is an AND ancestor of $J \}$.

But for AND nodes the negamax value $v(e)$ is the same as the minimax value $g(e)$.

$\alpha(J) = \max \{ -\min \{ g(e) \mid e$ is an elder brother of $a \} \mid a = J$ or $a$ is an AND ancestor of $J \}$

$= -\min \{ \min \{ g(e) \mid e$ is an elder brother of $a \} \mid a = J$ or $a$ is an AND ancestor of $J \}$

$= -\hat{h}(n)$
because all the nodes are part of the solution tree which \( SSS^* \) has traversed up to state \((J, \text{LIVE}, \hat{h})\). For the notation to be meaningful in case there are no elder brothers of node \( a \) we define \( \min \{ \} = +\infty \) just as we have defined \( \max \{ \} = -\infty \).

Lemma 2. If \( SSS^* \) explores AND node \( n \) then \( \hat{h}(n) > -\beta(n) \).

Proof: Suppose that \( J_j \) is an arbitrary OR node ancestor of node \( n \).

It must be that \( g(J_i) \leq \hat{h}(n) \) for \( i \neq j \). Suppose \( g(J_i) > \hat{h}(n) \) for some \( i \).

Then state \((J_i, \text{SOLVED}, \hat{h}_o)\) would have appeared at the top of OPEN with \( \hat{h}_o \geq \hat{h}(n) \). Thus \((J, \text{SOLVED}, \hat{h}_o)\) would have been the next state placed on top of OPEN and all states \((J_x, s, \hat{h})\) would have been purged from OPEN. Hence node \( n \) would not be explored by \( SSS^* \). Thus \( g(J_i) \leq \hat{h}(n) \) for \( i \neq j \) and clearly \( g(J_i) < \hat{h}(n) \) for \( i < j \), i.e. \( J_i \) is an elder brother of \( J_j \). Since \( g(J_i) \equiv -v(J_i) \) we have \( \hat{h}(n) > -v(J_i) \) for all \( i < j \) and for all OR ancestors \( J_j \) of node \( n \).

Thus \( \hat{h}(n) > \max \{ -v(J_i) \mid i < j \} \equiv c(J,j) \) for all OR ancestors \( J_j \) of node \( n \).

Hence \( \hat{h}(n) > \max \{ c(J,j) \mid J_j \text{ an OR ancestor of } n \} \equiv -\beta(n) \) as claimed. To touch up the proof in case \( j=1 \) and there are no elder brothers of \( J_j \) note that \( c(J,j) \equiv -\infty \) and that \( \hat{h}(n) \) being a minimum of \( +\infty \) and finite tip values, is never \( -\infty \).

4.3.7 Proof of \( SSS^* \) dominance.

It is now easy to conclude that \( SSS^* \) will explore fewer nodes than \( \alpha-\beta \).
Theorem 3 (Domination) If SSS* explores arbitrary node $n$ of a game tree then so must $\alpha-\beta$.

Proof: Lemmas 1 and 2 yield a trivial proof in case $n$ is an AND node. $\alpha(n) = -\hat{h}(n)$ by Lemma 1 and $-\hat{h}(n) < \beta(n)$ by Lemma 2. Thus $\alpha(n) < \beta(n)$ which from Baudets' theorem implies that $\alpha-\beta$ must explore node $n$.

Now assume that node $n=J.j$ is of type OR and explored by SSS*. SSS* must have explored the AND node parent $J$ and $\hat{h}(J.j) = \hat{h}(J)$ from Table I. In the proof of Lemma 2 it was shown that $\hat{h}(J) > c(J.j)$. By the definition of $\alpha$ and $\beta$ we have

$$\alpha(J.j) = \max \{ c(J.j), -\beta(J) \} \quad \beta(J.j) = -\alpha(J).$$

Now since $J$ is type AND and explored by SSS* it must be explored by $\alpha-\beta$ as well. Thus $\alpha(J) < \beta(J)$ or $-\beta(J) < -\alpha(J)$. Since $c(J.j) < \hat{h}(J) = -\alpha(J)$ it follows that $\max \{ c(J.j), -\beta(J) \} < -\alpha(J)$ and so $\alpha(J.j) < \beta(J.j)$. Again from Baudets theorem it follows that $\alpha-\beta$ must explore node $J.j = n$ and the proof is established.

It is now clear that SSS* is strictly superior to $\alpha-\beta$ in terms of nodes explored when reasonable probabilistic assumptions are made about tip value assignments. The following corollary gives an example.

Corollary: Let $f$ be an evaluation function assigning the $X$ tips of a game tree values from any nonsingleton set $R$ of real numbers. If each of the possible realizations of $f$ has non-zero probability, then SSS* can expect to explore fewer nodes than $\alpha-\beta$ on the average.
Proof: Let \( v_1 \) and \( v_2 \) be two values from \( R \) such that \( v_1 < v_2 \). Consider the leftmost set of terminal AND nodes \( \{ J \mid J = 1 \cdot 1 \cdot 1 \cdots \cdot 1 \cdot i \} \). \( \alpha\beta \) must explore this entire set of nodes regardless of their value. However, if the value of the leftmost node is \( v_1 \) and the value of the game tree is \( v_2 \) or greater \( \text{SSS}^* \) will explore only the leftmost node of the set. Thus \( \text{SSS}^* \) would explore strictly fewer nodes than \( \alpha\beta \) in this case and in no case explore more nodes by Theorem 3. By assumption, this case has non-zero probability and therefore the expected number of nodes evaluated by \( \text{SSS}^* \) would be less than for \( \alpha\beta \).

Just how much better \( \text{SSS}^* \) can do depends on the distribution of possible tip assignments and is left for future study. The simulation results give some indication for random distinct tip assignments.

4.3.8 The storage efficiency of \( \text{SSS}^* \)

Let the game tree consist of \( K \) levels of AND nodes and \( K \) levels of OR nodes beyond the root and let the branching factor be \( N \). Also assume that there is not a guaranteed win for the root player; i.e. \( g(1) < +\infty \). Because \( \text{SSS}^* \) starts with state \((1, \text{LIVE}, +\infty)\) and propagates \( N \)-fold with case 6 of \( \Gamma \) (Table I), OPEN will receive \( N^K \) states of merit \( \hat{h} = +\infty \) before \( \text{SSS}^* \) can possibly terminate with \( g(1) < +\infty \).

For instance, by state 13 of example 3.3 the fourth state of merit \( +\infty \) appeared at the top of OPEN. There can never be more than \( N^K \) states on OPEN because AND nodes will be "solved" at the frontier without increasing the size of OPEN, and sooner or later OR nodes above in the tree will become solved and OPEN will actually decrease in size. Requiring \( N^K \) OPEN entries, \( \text{SSS}^* \) is very inefficient in storage in comparison to \( \alpha\beta \) which requires only \( 2K \) stack entries for the same game tree.
5. Discussion and conclusions

This research originated not in game playing but in syntactic pattern recognition. The SSS procedure was constructed to produce ambiguous parses of waveforms modeled by a context-free grammar. Evaluation of terminals corresponded to curve fits of a mathematical model to a segment of waveform data. Terminal values were $\chi^2$ values of the confidence in the fit. The states of SSS were actually entire partial parse trees and expansion of the states successively enforced the syntactic constraints of the grammar. The best fitting parses were developed first by the best-first order of SSS. Since we were interested in multiple interpretations no states were removed from OPEN unless their value fell below a threshold. In addition SSS was often operated in a bottom-up mode by initializing OPEN with states (J, SOLVED, h) where J was a tip node and $\hat{h} = f(J)$. Despite quite large encodings of states the procedure was able to develop parses of significant pulse wave data in a few seconds with only a few thousand words of dynamic storage. Details are provided in [6] for the interested reader.

Knuth and Moore [2] halted their search for a better algorithm (on page 298) and devised an optimality theorem (page 307) for $\alpha-\beta$. The definition of optimality in that theorem seems rather sterile in light of the results reported here. Once again a classical tradeoff between storage space and execution time has been achieved. $\alpha-\beta$ is bound to sequential left-to-right development of the game tree while SSS* sinks multiple paths to the leaves and then competitively develops alternative traversals. $\alpha-\beta$ loses efficiency when the best solution is toward the right of the tree. If there exists accurate ordering information for successors
this loss will not occur because the best solution after reordering will be
toward the left of the game tree. It may turn out to be more practical to
use SSS*, however, than to order successors. SSS* may also be used to advan-
tage on a system with true parallel processing.

For a game tree of depth 2^K and branching factor N SSS* requires N^K cells
of storage for the OPEN list. This presents no practical problem only for small
N or K. The stack size of α-β varies only with depth and is independent of N so a
meager 2^K cells of storage are required. The steps of SSS* are slightly more
complicated than those of α-β and thus SSS* will be slower than α-β when identi-
cal node sets are explored. The complication arises in maintenance of the ordered OPEN list. Some insight can be gained by reexamining Table III Columns 8 and
10. SSS* was implemented as quickly as possible and was not designed to be effi-
cient according to any criteria other than number of tip nodes explored. SSS*
consistently ran slower than α-β in the simulations. However, tip evaluation time
was not considered. For instance, on the game trees with N = 2 and K = 2 SSS*
took 12 seconds longer but evaluated 2000 fewer tip nodes. If the terminal evalu-
uation function requires greater than 0.006 seconds, SSS* would in practice execute faster. The break even point is less than 0.002 seconds for N = 5 and K = 2. Re-
design of the OPEN list structure in the implementation of SSS* would further
enhance the practical comparison to α-β.

SSS* has been shown to be superior to α-β according to the criteria of number
of nodes evaluated. Simulation results indicate that SSS* can in fact execute
faster than α-β in certain practical cases. It seems likely that a hybrid SSS*-α-β
procedure would be worthwhile. α-β could be used to a certain depth to avoid SSS*
storage problems and then SSS* could be used effectively to save on tip evaluations. Alternatively SSS* could be used at the shallow level and α-β at deeper levels of the game tree. More work needs to be done in this regard. Hopefully other AI workers will further examine the tradeoffs involved and evaluate the potential of SSS* for their application. There are also theoretical questions left open. It is easy to see that SSS* has the same best case as α-β and thus is still an expensive search procedure even in the best situations. From the results of this paper any upper bound on α-β efficiency also applies to SSS*. Deep cutoffs are now known to have only a secondary effect on the character of α-β [1,3]. What is the effective branching factor of SSS* and does it have a character significantly different from α-β?
References


