MIPS-XXL: Featuring External Shortest Path Search for Sequential Optimal Plans and External Branch-And-Bound for Optimal Net Benefit [*]

Stefan Edelkamp and Shahid Jabbar
Faculty of Computer Science
TU Dortmund, Germany
{stefan.edelkamp,shahid.jabbar}@cs.uni-dortmund.de

Abstract

For large planning problems, the size of the state space can easily surpass the limits of main memory. External-memory algorithms exploit available disk space storing large state sets in files. In this paper, we describe changes to existing set-based exploration algorithms in MIPS-XXL that we have implemented to participate in IPC-6. Thereby, we describe changes for single-source shortest path search and for handling net-benefits.

Introduction

Search algorithms and their variants, play an important role in many branches of computer science. All use duplicate detection in order to recognize when the same node is reached via alternative paths in a graph. This traditionally involves storing already-explored nodes in random-access memory (RAM) and checking newly-generated nodes against the stored nodes. However, the limited size of RAM creates a memory bottleneck that severely limits the range of problems that can be solved with this approach. Although many clever techniques have been developed for searching with limited RAM, all eventually are limited in terms of scalability, and many practical graph-search problems are too large to be solved using any of these techniques.

Over the past few years, several researchers have show that the scalability of graph-search algorithms can be dramatically improved by using external memory, such as disk, to store generated nodes for use in duplicate detection. However, this requires very different search strategies to overcome the six orders-of-magnitude difference in random-access speed between RAM and disk.

Compared to the virtual memory management of the operating systems, search algorithms that exploit the memory hierarchy can lead to substantial speedups. These algorithms are more informed to localize the future memory accesses. Such algorithms are analyzed on External Memory (EM) model for their I/O complexities rather than on von Neumann architecture (Aggarwal and Vitter 1988). The EM Model consists of a single processor, a small internal memory that can hold up to M data items, and an unlimited secondary memory. The size of the input problem (in terms of the number of data items) is abbreviated by N. Data is transfered between the internal memory and the disk in blocks of sizes B. Only the number of block reads and writes are counted, computations in internal memory do not incur any cost. In the EM Model, the available amount of external memory is infinite.

In this paper, we mainly consider optimal STRIPS planning (Fikes and Nilsson 1971) with costs as proposed for one track in the sixth international planning competition IPC-6, even though metric quantities are handled by our planner. For a given planning problem $\mathcal{P} = (S, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{C})$, the task is to find a sequence of actions $a_1, \ldots, a_k \in \mathcal{A}$ from $\mathcal{I}$ to $\mathcal{G}$ with minimal $\sum_{i=1}^{k} \mathcal{C}(a_i)$. We assume discrete costs and that the highest action cost is bounded by some constant C. Later on, we look at preferences in form of soft constraints and optimal planning for net-benefits.

External Shortest Paths

The earliest reference[1] to a disk-based Breadth-First Search (BFS) dates back to Munagala and Ranade's algorithm (1999) for exploring an explicit graph (graph provided in the form of adjacency lists). They proposed to represent each BFS-layer with a buffered file. Since, a hash table for large graphs cannot fit in the main memory, they proposed a layered duplicate removal method based on disk-based sorting and set subtraction. For implicit graphs (provided in the form of an initial state and a set of transformation rules), the algorithm was adapted and termed as *delayed duplicate detection* for *frontier search* by Korf (2003).

In large-scale optimal planning on graphs where costs are associated with actions, a simple EM-BFS cannot guarantee that the first plan found is an optimal one. A natural choice is to adapt the famous single-source shortest-path algorithm of Dijkstra (1959) for External Memory setting. The main data structure used by Dijkstra's algorithm is a priority queue. For an EM Dijkstra's algorithm on large implicit graphs, we propose to simulate the priority queue through a set of files. Let *Open*$[i]$ be the set of states reachable from the initial state through a shortest path of length i. Each *Open*$[i]$ is represented by a file on the hard disk. When *active* (being

[*]Thanks to DFG for support in ED 74/4.
Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]There is some earlier work by (Stern and Dill 1998) but this is not analyzed in the EM Model.

Algorithm 1 External Memory Shortest Path Algorithm

```
Input: $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{C})$
Output: Optimal solution path, if exists; error otherwise.

$Open[0] \leftarrow \mathcal{I}$
for all $f = 0, 1, 2, \ldots$
    $Open[f] \leftarrow SortAndRemoveDuplicates(Open[f])$
    for all $l = 0, \ldots, f-1$
        $Open[f] \leftarrow Open[f] \setminus Open[f-l]$
    if $(Open[f-C] \cup \ldots \cup Open[f] = \emptyset)$
        return "Exploration completed, no plan found"
    if $(Open[f] \cap \mathcal{G} \neq \emptyset)$
        return $ConstructPlan(Open[f] \cap \mathcal{G})$
    for all $i = 1, \ldots, C$
        $Succ_i \leftarrow \bigcup_{a \in \mathcal{A}, \mathcal{C}(a)=i} \{v \in \Gamma(u,a) \mid u \in Open[f]\}$
        $Open[f+i] \leftarrow Open[f+i] \cup Succ_i$
```

read or written to), a small memory buffer is also associated with the set *Open*$[i]$.

The external shortest-path search procedure is implemented in Algorithm 1. File *Open*$[0]$ is initialized with the initial state. Unless the goal is reached, we iteratively choose the next f-value from teh priority queue, such that *Open*$[f] \neq \emptyset$. For each $a \in \mathcal{A}$ with $\mathcal{C}(a) = i$ the successor function $\Gamma(u, a)$ is applied to all states $u \in$ *Open*$[f]$ to generate a set of states reachable with cost $f + i$. All such states are collected in the state set *Open*$[f + i]$. Before expanding a state set, we need to remove all the states that have previously been expanded from *Open*$[f]$. This step is done in two phases. First, *Open*$[f]$ is sorted through an external sorting algorithm, to bring all the duplicate states adjacent to each other. The set is then scanned to remove all the extra copies of a state. In the second phase, all the previously expanded state sets *Open*$[0] \ldots$ *Open*$[f - 1]$ are subtracted from *Open*$[f]$, resulting in *Open*$[f]$ to be duplicates-free.

As actions with a cost value of zero are allowed in the domain definition, the above mentioned algorithm has to incorporate the generation of successors in the same file. We chose to adapt the same approach as proposed for the symbolic planner, MIPS-BDD. Upon encountering an action with a zero cost, we successively apply all the zero-cost actions on the successors until a fix-point is reached. This step results in all states collected in a set that are reachable by a non-zero-cost action and applying one action with non-zero cost followed by a sequence of zero-cost actions.

Locality in Planning Domains

For undirected graphs, Munagala and Ranade (1999) suggested that it is sufficient to subtract previous two layers to remove all the duplicates. For general directed and weighted graphs though as we see in optimal STRIPS planning, the sufficient number of layers needed to subtract is dependent on a property of the graph called *locality*. Let *Layer*(u) be the correct total cost layer for a node u and $\Gamma(u) = \cup_{a\in\mathcal{A}}\Gamma(u, a)$ be the successors of a state u. The locality of a directed graph G with a weight function $\mathcal{C}$ is

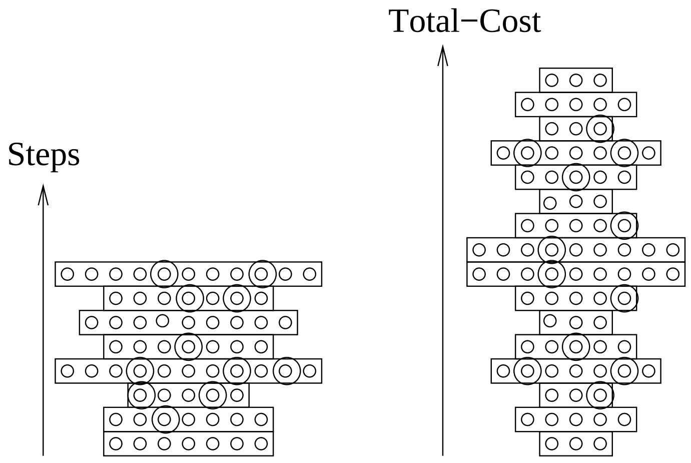

Figure 1: Breadth-First and Cost-First Branch-and-Bound Layers.

defined as

$$locality_G = \max_{u,v \ s.t. v\in\Gamma(u)} \{Layer(u) + Layer(v) + \mathcal{C}(u, v)\}$$

For breadth-first search graphs organized wrt. total cost, the locality determines the number of previous layers that need to be subtracted to guarantee that no node is expanded twice. Figure 1 illustrates that the search space is stretched along the benefit (files that are generated in the search process are filled with states, goal states are highlighted with a second circle).

This band of layers is usually termed as *Duplicate Detection Scope* and is the basis of the following lemma due to (Zhou and Hansen 2006). As the original proof in (Zhou and Hansen 2006) is not rigorous, we provide an alternative one and extend it to general weighted graphs.

Lemma 1 *(Locality Determines Duplicate Detection Scope) In a directed and weighted graph G, the number of previous layers of a breadth-first search graph that need to be retained to prevent duplicate search effort is equal to the locality$_G$.*

Proof: Let us consider two nodes u and v, with $v \in \Gamma(u)$. Assume that u has been expanded for the first time, generating the successor v which has already appeared in the layers $0, \ldots, Layer(u) - locality_G$ implying $Layer(v) \leq Layer(u) - locality_G$. We have

$$\begin{aligned} locality_G &\geq Layer(u) - Layer(v) + \mathcal{C}(u, v) \\ &\geq Layer(u) - (Layer(u) - locality_G) + \mathcal{C}(u, v) \\ &= locality_G + \mathcal{C}(u, v) \end{aligned}$$

This is a contradiction to $\mathcal{C}(u, v) > 0$. □

Since in undirected graphs *Layer*(u) − *Layer*(v) has a maximum value of 1, given that the graph is also unweighted, we only need to subtract previous two layers.

Given that C is the maximum edge weight that appears in the graph, the locality can be bounded.

Lemma 2 *(Upper-Bound on Locality) The locality of a uniformly positively weighted graph is bounded by the minimal* *distance $\delta(v,u)$ from a successor node v back to u, maximized over all u. In other words, we have*

$$\max_{u,v\in\Gamma(u)}\{\delta(v,u)\}+C \geq \max_{u,v\in\Gamma(u)}\{\mathit{Layer}(u)-\mathit{Layer}(v)\}+C$$

Proof: For any nodes u,v in a graph, the triangular property of shortest paths $\mathit{Layer}(u) \leq \mathit{Layer}(v) + \delta(v,u)$ is satisfied. Therefore, $\delta(v,u) \geq \mathit{Layer}(u) - \mathit{Layer}(v)$ and $\max_{u,v\in\Gamma(u)}\{\delta(v,u)\} \geq \max_{u,v\in\Gamma(u)}\{\mathit{Layer}(u) - \mathit{Layer}(v)\}$. In positively weighted graphs we have $\delta(v,u) > 0$ such that $\max_{u,v\in\Gamma(u)}\{\delta(v,u)\} + 1$ is larger than the locality. □

The question then arises is: How can we decide the compute the locality in an implicitly given graph as they appear in action planning? In general the answer is "no, we cannot". Nevertheless, in special cases, we can exploit the structure of the graph to compute an upper bound on the locality. For the ease of presentation, we restrict to actions with cost 1. A duplicate node in an implicit graph appears when a sequence of operators, applied to a state generate the same state again, i.e., they cancel the effects of each other. Hence, the following definition:

Definition 1 *(*no-op *Sequence) A sequence of operators $a_1, a_2, \ldots, a_k$ is a* no-op sequence *if its application on a state produces no effects, i.e, $a_k \circ \ldots \circ a_2 \circ a_1 =$* no-op*,*

This definition allows to bound the locality in the following proposition. It generalizes the observation that for undirected search spaces, in which for each operator a_1 we find an inverse action a_2 such that $a_2 \circ a_1 = \textit{no-op}$.

Proposition 1 *(*no-op *Sequence determines Locality) Let $\mathcal{A}$ be the set of operators in the search space and $l = |\mathcal{A}|$. If for all operators a_1 we can provide a sequence $a_2, \ldots, a_k$ with $a_k \circ \ldots \circ a_2 \circ a_1 =$* no-op*, then the locality of the implicitly generated graph is at most $k-1$.*

Proof: If $a_k \circ \ldots \circ a_2 \circ a_1$ from each state v reached by a_1 we can reach each state u again in at most $k-1$ steps. This implies that $\max_{u,v\in\Gamma(u)}\{\delta(v,u)\} = k-1$. Theorem 2 shows that this imposes the stated upper bound on the locality. □

It suffices to check that the cumulative add effects of the sequence is equal to the cumulative delete effects. Using the notation by (Haslum and Jonsson 2000), the cumulative add σ_A and delete σ_D effects of a sequence can be defined inductively as, $\sigma_A(a_k) = A_k$, $\sigma_D(a_k) = D_k$, and

$$\begin{aligned}\sigma_A(a_1,\ldots,a_k) &= (\sigma_A(a_1,\ldots,a_{k-1}) - D_k) \cup A_k\\ \sigma_D(a_1,\ldots,a_k) &= (\sigma_D(a_1,\ldots,a_{k-1}) - A_k) \cup D_k\end{aligned}$$

This result gives us the missing link to the successful application of external breadth first search in planning. Subtracting k previous layer *plus* the current layer from the successor list in an external breadth-first search guarantees its termination on finite planning graphs.

Net Benefit

In planning with preferences, we often have a monotone decreasing instead of a monotonic increasing cost function to be minimized. Hence, we cannot prune states with an evaluation larger than the current one. Essentially, we are forced to look at all states.

The branch-and-bound algorithm we have developed in the context of the fifth international planning competition incrementally improves an upper bound U on the solution length. When it comes to analyzing a layer in which more than one goal is contained a goal g with the minimum value $\mathcal{C}(g)$ is selected for solution reconstruction.

Net-benefit planning at IPC-6 concerns trading utility received by achieving soft goals for total cost of the actions used to achieve them. Planners competing in the optimal track are expected to find best plans in terms of a linear objective function. Note that maximization problems (as appearing in the IPC-6 benchmark suite) can be easily transformed into minimization problems (as described here), and adding a constant offsets (also used in the benchmarks for judging sub-optimal planners) does not change the solution set.

We consider optimal planning with action costs, metric quantities and goal utilities. For preference constraints of type $(\textit{preference}\ p_i\ \phi_i)$ we associate a Boolean variable $\textit{violated}_i$ (denoting the violation of constraint p_i). For the sake of brevity, we assume no scaling of the total cost value and coefficients α_i of the variables $\textit{violated}_p$ are expected to be discrete. Let $\textit{benefit}(\pi) = \textit{benefit}(s_n) = \sum_i \alpha_i \textit{violated}_i(s_n)$. Note that the notation of benefit has been inverted, it decreases with the satisfaction of the constraints.

By modifying domain actions, MIPS-XXL is capable of compiling the PDDL3 Boolean variables $\textit{violated}_i(s_n)$ into ordinary PDDL2 fluents. Then the metric we consider is $\textit{net-benefit} = m = \textit{benefit} + \textit{total-cost}$.

As above, state sets that are used are represented in form of files. The search frontier denoting the current layer is tested for an intersection with the goal, and this intersection is further reduced according to the already established bound. The pseudo-code of the algorithm is shown in Figure 2.

Fortunately, while expanding a state set $\textit{Open}[f]$, we already have the current f-value for evaluating *total-cost* at hand. This allows to use a different upper bound in the branch-and-bound algorithm. The pseudo-code is presented in Algorithm 2. With V we denote the current best solution obtained according to m, which improves over time. With V' we denote the old value of V. As the f-value increases monotonically, we can also adapt V to improve over time. The pseudo-code also adapts a small refinement, by observing that the upper bound U for $\textit{benefit}(\pi)$ is bounded V, which is effective if the impact *total-cost* is small.

Conclusion

We have shown how to extend MIPS-XXL to compute plans with total action cost and net benefit. Recall that MIPS-XXL is based on Metric-FF and can compute plans with fluents. In contrast to the IPC-6, we participate in the optimal track and not the suboptimal track. MIPS-XXL also no more invoke an internal memory planning algorithm before starting the external memory algorithm.

Algorithm 2 External Net-Benefit Planning Algorithm.

```
Input: Problem $\mathcal{P}$ with action cost $\mathcal{C}(a)$, $a \in \mathcal{A}$, Cost
       function $m = \textit{benefit} + \textit{total-cost}$ to be minimized
Output: Cost-optimal plan from $\mathcal{I}$ to state satisfying $\mathcal{G}$

$U \leftarrow \max_{\textit{benefit}}$; $f = \min_{\textit{benefit}} + 1$
$V \leftarrow V' \leftarrow \infty$
$\textit{Open}[f] \leftarrow \mathcal{I}$
loop
for all $f = 0, 1, 2, \ldots$
  $\textit{Open}[f] \leftarrow \textit{SortAndRemoveDuplicates}(\textit{Open}[f])$
  for all $l = 0, \ldots, f-1$
    $\textit{Open}[f] \leftarrow \textit{Open}[f] \setminus \textit{Open}[f-l]$
  if $(\bigcup_{i=f-C}^{f} \textit{Open}[i] = \emptyset$ or $(U = \min_{\textit{benefit}})$
    return $\textit{StoredPlan}()$
  if $(\textit{Open}[f] \cap \mathcal{G} \neq \emptyset)$
    $U' \leftarrow \min_{\textit{benefit}}$
    while $U' + f < V \wedge \exists v \in \textit{Open}[f] \cap \mathcal{G}.\textit{benefit}(v) = U'$
      $U' \leftarrow U' + 1$
    if $(U' + f < V)$
      $V \leftarrow U' + f$
      if $(V - f < U)$ $U \leftarrow V - f$
      if $(V < V')$
        $V' \leftarrow V$
        $\textit{Construct\&StorePlan}(\textit{Open}[f] \cap \mathcal{G}, U')$
  for all $i = 1, \ldots, C$
    $\textit{Succ}_i \leftarrow \bigvee_{a \in \mathcal{A}, \mathcal{C}(a)=i} \{v \in \Gamma(u,a) \mid u \in \textit{Open}[f]\}$
    $\textit{Open}[f+i] \leftarrow \textit{Open}[f+i] \cup \textit{Succ}_i$
```

Unfortunately, we do not feature functional STRIPS that has been proposed in the context of IPC-6 as an extension to PDDL. The core reason is that no instantiation mechanism is available yet. For external planning, the functional representation can result in a smaller state vector. However, it might not be considered a drastic change.

In difference to the previous heuristic search (Edelkamp and Jabbar 2006), and breadth-first branch-and-bound algorithms (Edelkamp, Jabbar, and Nazih 2006), due to the monotonicity of the cost function in IPC-6, not all states are looked at.

As a side effect, the branch-and-bound algorithm also solves the shortest paths problem. Hence, to avoid too many files processed for large values of C, we may restrict to breadth-first branch-and-bound in the competition.

The two algorithms proposed are not much different from the BDD exploration algorithms that we have developed in the context of IPC-6. Up to the elimination of duplicates and the computation of the image, the algorithms are pretty much the same. This again validates the hypothesis that the core design technique for joint symbolic and external algorithms is to derive set-based algorithms.

References

Aggarwal, A., and Vitter, J. S. 1988. The input/output complexity of sorting and related problems. *Journal of the ACM* 31(9):1116–1127.

Dijkstra, E. W. 1959. A note on two problems in connection with graphs. *Numerische Mathematik* 1:269–271.

Edelkamp, S., and Jabbar, S. 2006. Cost-optimal external planning. In *AAAI*, 821–826.

Edelkamp, S.; Jabbar, S.; and Nazih, M. 2006. Large-scale optimal PDDL3 planning with MIPS-XXL. In *Proceedings of the International Planning Competition. International Conference on Automated Planning and Scheduling*, 81–92.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Haslum, P., and Jonsson, P. 2000. Planning with reduced operator sets. In *AIPS*, 150–158.

Korf, R. E. 2003. Breadth-first frontier search with delayed duplicate detection. In *Model Checking and Artificial Intelligence (MOCHART)*, 87–92.

Munagala, K., and Ranade, A. 1999. I/O-complexity of graph algorithms. In *SODA*, 687–694.

Stern, U., and Dill, D. 1998. Using magnetic disk instead of main memory in the murphi verifier. In *International Conference on Computer Aided Verification (CAV)*, 172–183.

Zhou, R., and Hansen, E. 2006. Breadth-first heuristic search. *Artificial Intelligence* 170:385–408.