Extending AFB - BackJumping

In this chapter we present the AFB-BJ algorithm. AFB-BJ is an extension of the AFB algorithm that incorporates a backjumping mechanism for improved performance. Sometimes during the run of the AFB search, the CPA is backtracked to some agent $A_i$ after it was determined to be a dead end. $A_i$ then attempts to replace its assignment with another assignment, and if successful passes the CPA to the next agent $A_{i+1}$. $A_i$ would repeat this process whenever the CPA is backtracked from $A_{i+1}$, until it has attempted to extend the CPA with every value in its domain of values, and only when it has tested them all would it backtrack the CPA to $A_{i-1}$. The proposed mechanism for backjumping allows agents such as $A_{i+1}$ to detect situations in which the CPA can be sent directly to agents prior to $A_i$ without missing out on potential solutions. As can be seen from the above description, this avoids useless computation, as $A_i$ explores its remaining values.

Backjumping is widely used in CSP algorithms. The conflict-based backjumping algorithm (CBJ) \[51\] is a proven example of a speed up mechanism for centralized CSPs \[14\]. In CBJ, conflict sets hold the culprit variables responsible for the elimination of values from each variable’s current domain. When backtracking is required, these conflict sets can be used to identify the culprit variables responsible for the dead end, and therefore allow backjumping directly to the latest variable that was assigned among those variables. Graph-based backjumping \[16\] is another form of backjumping, in which backjumping possibilities are inferred from the constraint graph.

Recently, conflict-based backjumping was also applied to MAX-CSPs \[76\]. The extension of backjumping to optimization problems is not trivial, since culprit variables are harder to detect. When accumulating explanations for eliminated values in constrained optimization problems, many values involved in some constraints might be part of the optimal solution. Bounds for each value as well as the current lower and upper bounds must be used to detect the culprit variables. To the best of the author’s knowledge, no other DiscOP algorithm to date uses any form of backjumping mechanism.
17.1 Adding Value Ordering Heuristics

Before adding a backjump mechanism, one first needs to add a value ordering heuristic. A value ordering heuristic is a heuristic for reordering the domain values of agents. Since agents pick their next assignment to be the next untested value in their value domain, ordering these values in different ways has the potential to produce different performances, as shown for example in [14]. A good value ordering heuristic for AFB is called \textit{min-cost} and the resulting version of the algorithm is called accordingly AFB-\textit{minC}. The \textit{min-cost} heuristic arranges the values of an agent by the cost of each value with respect to the assignments of higher priority agents on the CPA. Each agent performs this reordering whenever it needs to perform an assignment. Since the ordering cannot change without a change in the assignment of higher priority agents, no reordering is performed as long as these assignments remain fixed. Once an updated CPA arrives, containing new assignments for higher priority agents, the agent reorders its values and the ordering remains until the next time higher priority agents change assignments (in other words, until a backtrack is performed).

Since AFB does not assume any special ordering of the values, it remains correct and complete with any specific ordering. To see why the algorithm remains correct, observe that an obsolete ordering is in a one-to-one correspondence to a time stamped CPA, i.e., to an assignment of higher priority agents. Any message received that is related to an obsolete ordering and is discarded is also related to an obsolete time stamped CPA. As long as the current CPA remains valid, the value ordering is fixed and there is no possibility of missing exploration of values due to reordering during search.

17.2 Backjumping - Key Concepts

In both centralized and distributed CSPs, backjumping can be accomplished by maintaining data structures that will allow an agent to deduce who is the latest agent (in the order in which assignments were made) whose changed assignment could possibly lead to a solution. Once such an agent is found, the assignments of all following agents are unmade and the search process backjumps to that agent [23, 44, 51].

A similar process can be designed for branch and bound based solvers for COPs and DisCOPs. Consider a sequence of assignments by the agents $A_1, A_2, A_3, A_4, A_5$ where $A_5$ determined that none of its possible value assignments can lead to a full assignment with a cost lower than the cost of the best full assignment found so far. Clearly, $A_5$ must backtrack.

In chronological backtracking, the search process would simply return to the previous agent, namely $A_4$, and have it change its assignment. However, $A_5$ can sometimes determine that no value change of $A_4$ would suffice to reach a full assignment of a lower cost. Intuitively, $A_5$ can safely backjump to $A_3$, if