NB^3: Negotiation-Based Branch&Bound
TR—IIIA—2011–03

Carles Sierra
Artificial Intelligence Research Institute, IIIA-CSIC
Campus de la Universitat Autònoma de Barcelona
sierra@iiia.csic.es

Abstract

In this paper I introduce a new multiagent negotiation algorithm that explores the space of joint plans of action: NB^3. Each negotiator generates a search tree by considering both actions performed by itself and actions performed by others. The algorithm prunes the nodes of the tree that require rejected actions of others, and focusses on the most promising nodes by using appropriate heuristics.

1 Introduction

Negotiation algorithms have been frequently used to co-ordinate autonomous agents. Although negotiation has been rightly described as a search problem [2], previously proposed negotiation algorithms have mostly focused on the utility space, that is, on how the utility value of the proposals and counterproposals changed along time and how they approached the pareto-optimal frontier. In a sense, these algorithms assume that given a utility aspiration level or a concession degree it is always possible to find a proposal that would fit that degree. This might be true given certain continuity assumptions (e.g. variables with real values) but is often not the case when the domains of the issues are discrete or when there are integrity constraints among them [11]. In this paper I focus on complex problems for which the classical continuity assumptions do not apply and thus solutions have to be directly found at domain level. Also, I address a number of realistic assumptions that make the application of current negotiation algorithms unfeasible:

- The space of solutions is huge, i.e. there is no possibility to exhaustively explore the set of solutions.
- Solutions improve with co-operation. Some actions are interdependent, if agents help each other they are individually better off.
- The environment is only partially observable by each agent, e.g. actions made by others my not be observable.

- The environment changes due to actions of others.
- Decisions have to be made within a limited time frame.
- Solutions may involve a large number of agents, possibly including humans.

Many difficult problems belong to this class, e.g:
Time Tabling: School teachers have private goals for their schedule. Once a schedule is determined by the School Master, they may improve their particular allocations by negotiating local exchanges with fellow teachers. This can be based on bilateral or multilateral negotiations. Teachers' goals here account, for instance, for compactness (avoiding 'holes' in their schedules), for compatibility with other activities (allowing a couple of hours in the middle of the schedule to practice sport), or for personal taste (avoid afternoons as kids get too excited). Current software solutions [10] are centralised and do not permit negotiation among teachers.
Diplomacy: Diplomacy is a classical board game where seven players incarnate major European powers at the beginning of the XXth Century. They own territories and have to conquer half of Europe in order to win. There are no chance moves and the only way to progress in the game is to get the support of other powers. Good negotiators win. Developed software bots [6] do not offer any reasonable negotiation techniques and are thus vulnerable when playing with humans that show great capacity in negotiation [4]. The search space is in this case colossal.
Logistics: A key issue in logistics is how to optimise multi-truck scheduling of package delivery between companies where every connection between nodes in the delivery network has a cost (petrol + time) and every package delivery has a price. Current centralised systems [5] are not reactive enough to dynamic changes (e.g. new deliveries constantly appearing, road incidents) and call for a more efficient distributed solution where the negotiation of who transports what is done at the truck's and customer's level allowing for private preferences to enter into scene (a driver may prefer a detour to pass by his house and thus be keen on accepting a low profit delivery). A simplified version of this problem where postmen exchanged letters to minimise their individual costs was studied in [8].

In this paper I introduce a new family of Branch and Bound algorithms, namely NB^3, that use negotiation as the key element in the exploration of the joint space of solutions for a number of autonomous agents. Section 2 briefs on Branch and Bound (BB) algorithms. Section 3 explains the concept of the algorithm. Then, in Section 4 I give the details of the algorithm whose main heuristic is described in Section 5. Finally, in Section 6 I conclude.

2 Branch & Bound algorithms in a nutshell

Branch&bound (BB) is a general algorithm to find optimal solutions in discrete domains. Here I outline the basics of the algorithm and introduce some notation, for an in-depth description refer to [3, 7].

The objective of a *BB* algorithm is to find a solution x to a problem that maximises (or minimises) a given function $f(x)$. The algorithm incrementally generates a tree where nodes represent sets of solutions S. Children nodes represent subsets of the father $(S_1, \ldots, S_i, \ldots, S_n)$ forming (ideally) a partition. A *BB* algorithm consists of three basic operations. One to split the set of solutions represented in a node into a number of subsets that become the children of the node. This operation is called *Branching*. It is clear that

$$max_{x \in S} f(x) = max\{v_i \mid v_i = max_{y \in S_i} f(y)\}$$

The second operation is the establishment of bounds, lower and upper, for the value of $f(x)$ on the elements of S_i. This step is called *Bounding*. The key idea in any *BB* algorithm that looks for a maximum of $f(x)$ is that if the upper bound of a node is lower than the lower bound of another node, then the former node can be ignored as it will not contain the optimal solution. This third step is the *pruning* of the tree. This recursive procedure stops when the set S contains a single element or when the lower and upper bounds get equal, i.e. all contained solutions are equally good (or bad).

The efficiency of the algorithm heavily depends on how well the splitting and bounding procedures are done. As guidelines, the smaller the overlap among children the better, and the more accurate the bounding the better. Without a good bounding there will be little pruning and the tree will become an exhaustive search of the space of solutions which is usually impractical. An advantage of the algorithm is that, as it progresses, the global bounds, defined (in case of minimisation) by the minimum of the lower and the minimum of the upper bounds of the non-pruned nodes, gets reduced. We can also stop the search, i.e. the splitting of a node, when its interval gets reduced to a reasonable size and then pick up one element randomly from the set represented by the node.

3 NB^3 basic concept

Branch and bound has been mostly used as a centralised algorithm. Distributed versions do also exist that try and exploit concurrency in the exploration of the tree [1]. However, not much work has been done on the application of branch&Bound algorithms in search problems where the splitting is based on variables that are controlled by different entities (i.e. agents), as in the asynchronous backtracking method used in Distributed Constraint Satisfaction [11], and where there is no single function $f(x)$ to optimise but a *set* of functions, one per agent, that are not centrally known. This paper proposes an algorithm that is run by every agent in a multiagent system (its execution could also be distributed, but this is not the relevant focus here) and that uses negotiation between agents to split nodes and to prune nodes that contain sets of undesirable solutions (worst than others already found) or unfeasible solutions (they require actions explicitly rejected by some agents). Next I give the basic idea of the algorithm.

I assume a number of agents $A = \{\alpha, \beta, \ldots, \omega\}$ situated in an environment $\epsilon \in \mathcal{E}$. Each agent α has the capacity of executing a set of feasible actions in a particular environment and given a set of known commitments, $\mathit{fea}(C_\alpha, \epsilon) \subset \mathcal{O}$, where $\mathcal{O}$ represents the set of all possible actions by any agent in any environment and $C_\alpha \subseteq \mathcal{O}$. For convenience, inaction is considered as a possible action. At particular time instants, agents decide autonomously what action (or actions) to perform in the environment. They are endowed with private goals and thus select those actions that might be more profitable for their goals. I am assuming environments where dependencies between actions are fundamental. In other words, certain actions performed by α will only be successful if they are accompanied by certain actions performed by β. This means that agents cannot decide what to do in isolation. They are assumed to have the capability of persuading one another, via negotiation, in order to co-ordinate their actions. I don't assume any global goal, agents use their private goals to determine the success of the agreed upon joint set of operations over the environment. I do assume that agents know which other agents are available and their possible actions. A set of actions, that is, a joint plan, $p = O_\alpha \cup O_\beta \cup \cdots \cup O_\omega$, where $O_i \subseteq \mathit{fea}(C_i, \epsilon)$ for all $i \in A$, executed on an environment ϵ will end up in a new environment ϵ', noted $p(\epsilon) = \epsilon'$. Agent α will measure how good ϵ' is to its goals with the help of a private function $f_\alpha(\epsilon)$. Instead of a single function to optimise $f(\epsilon)$, as in classical BB, in a multiagent setting we are then dealing with a tuple of functions $\langle f_\alpha(\epsilon), f_\beta(\epsilon), \ldots, f_\omega(\epsilon)\rangle$ each one being locally optimised by a copy of the NB^3 algorithm.

During the process agents make commitments to perform certain actions by accepting proposals and reject actions by rejecting proposals. Agents have a partial view of the commitments made as conversations may be private.

I next explain the different components of NB^3 from the perspective of agent α.

3.1 Search tree

In a multiagent setting, each agent that runs the algorithm builds its own search tree. The root node of the NB^3 search tree consists of all the possible solutions to the problem. In our context, assuming that each agent may chose to perform a subset of its feasible actions, the set is $S = \bigcup_{i \in A} \mathit{fea}(C_i, \epsilon)$. The set of plans represented by a node n, i.e. subsets of S, are noted as $\mathit{plans}(n)$. As already mentioned, this is in most practical applications an intractably large set for exhaustive exploration. The children of a node will be, as in the classical case, a partition of the solutions of the father node. I label the link between a father and a son with either (i) the name of an action contained in all the solutions represented by the child or (ii) with a constraint satisfied by all the solutions of the child. A path between the root and a leaf of the tree is then a (joint) (partial) plan (i.e. those actions labelling links in the path) that guarantees, at least, the worst solution in the leaf node. Given the path from node n backwards to the root node, I note by $n.\mathit{path}$ the set of actions in the path from all agents in A, that is $n.\mathit{path} \subseteq \bigcup_{i \in A} \mathit{fea}(C_i, \epsilon)$.

I also assume that α is situated in an environment that has strict time limits for a decision to be made. If the tree has been completely explored and an optimal solution has been found before the deadline then the decision of what α has to do is easy: the actions corresponding to α in the path to the optimal leaf node. Otherwise, the set of actions in the path from the root to the node in the partially explored tree with the best bounds plus the known commitments is a possibly good choice for action, even if only partial, and is what NB^3 considers as the best plan. In that respect, NB^3 is an anytime algorithm that always has the so far *best* plan of action ready.

3.2 Splitting

I consider two broad classes of actions that can be used to split a node:

- *Atomic actions.* An atomic action is an indivisible action on the environment. The labels of the arcs between a father and a set of children are *mutually exclusive atomic actions* generating a partition. The splitting will generate as many children as mutually incompatible actions. For instance, if α is considering the possible moves of one of its units (name given to a kind of 'army' in Diplomacy) and there are three mutually exclusive movements, three children would be generated.
- *Generic actions.* The labels of the arcs between a father and its children are *mutually exclusive generic actions* that generate a partition. The subset of solutions represented by each child satisfy the constraint in the arc back to its father. A generic action is a *behaviour agreement* between α and some other agent β and does not imply any concrete action to be made but a restriction/commitment on the actions that both agents will make.[1] Different semantics could be possibly associated to the same constraint by different agents.[2]

For simplicity, I consider that both generic and atomic actions are part of the set $\mathcal{O}$. Any generic or atomic action involving β and labelling an arc will force the issuing of an offer from α to β if the child node is at some point selected as the best current node, as it requires the acceptance of the terms of the agreement by β. I am assuming a negotiation environment in which commitments among the participants have to be made along the search process. This is so because an agent cannot wait until it finds the optimal plan before negotiating with other agents, as then it would be perhaps too late to get any commitment from them: they might have already signed commitments for incompatible actions. Therefore, a trade-off exists between optimality and commitment availability.

[1]An example in Diplomacy could be to do a binary split depending on whether a peace treaty is agreed upon with another agent or not. A peace treaty is not an *actual* action on the environment, but a constraint on what actions are allowed: a peace treaty between two agents prevents them from attacking each other, for instance.

[2]For instance, what actually peace means may be different: just not attacking or also helping if requested.

The more we forward explore from a potential commitment of others (without actually getting the commitment) the better, but then the less probable it is to get it. How to solve this trade-off is key in the application of NB^3 to a particular problem, NB^3 has an eager strategy: when a node is selected as the best, the commitments of other agents needed in the path to the node are immediately sought for.

Given that we have two types of splitting mechanisms and the mentioned trade-off, α needs to carefully find out the type of split: constraining or selecting, and which constraint or actions to use in the split. On the one hand, it may be good to initially split according to the actions α has full control (i.e. its own actions) as it can progress quickly in the expansion of the tree, but at the same time α needs to start negotiating for good enough commitments of the others as soon as possible (to prevent undesirable agreements among them). The heuristic h that proposes the split to make in a node is a fundamental parameter of NB^3. It ranks the splits to make at each node according to the path to the node $n.path$ and both the goals and the trust[3] attitude of α.

3.2.1 Offers

When a node is selected as the best current node, the agreements and actions in the path to the root that require commitments from other agents that have not yet been asked for makes the agent (i) to issue as many offers as needed to get them, and (ii) to withdraw any standing offers that are incompatible with the offers just made.

In particular, when the heuristic h used by α chooses to split a node according to β's actions or a constraint on a mutual agreement and, after bounding, one of the children is selected as the current best, NB^3 issues an offer to β to get its commitment on the action or agreement labelling the arc from the child to the father. If there is a standing offer that is incompatible with that action, NB^3 withdraws it.

Any offer to be sent by α should normally include those actions in the path to the root that α considers to be profitable to the involved partner(s). What to include is however part of the negotiation strategy of the agent and beyond the focus of this paper.

While waiting for the acceptance of issued offers, NB^3 keeps on expanding the tree. h should prioritise, when expanding the tree from a node, those actions of α that might be interesting to the agents with open negotiation threads in preparation for a counter-offer.

When α receives an offer from β, h should prioritise those actions contained in the offer that are compatible with those in the path to the current best node. In this way, agents help each other in focusing the search on the space of potential deals. Another key parameter of NB^3 is for how long should α explore before accepting an offer, as β may withdraw the offer if α waits too much.

[3]Don't forget that commitments may not be respected.

3.2.2 Withdraws

When α receives a withdraw from agent β it will prune all nodes that require those actions of β in the proposal withdrawn. Also, any withdraw is an indication that the probability of reaching a deal with β is lower than before and consequently the actions by β should have less priority in future selections to be made by h.

When α withdraws a previously sent offer to β the probability of reaching agreements with β decreases as β might be unhappy with the decision[4] and then the selection of actions that were potentially favourable to β and that h was going to select early in the splitting process should be delayed.

3.3 Bounding

Given an environment $\epsilon \in \mathcal{E}$, every time a node n is created, NB^3 generates its bounds: $n.min$ and $n.max$. The bounds are estimations of the minimum and maximum value of the environments ϵ', that may be generated from the joint plans represented in the node:

- *min*: an estimate of the value of the worst environment reachable from the joint plans in n. The worst environment is theoretically the one where the rest of agents coalesce to play against α. Which combination of actions is the worst to α may be difficult to estimate. The concrete function to use is domain dependent and must satisfy $n.min \leq min\{f(p(\epsilon)) \mid p \in plans(n)\}$

- *max*: an estimate of the value of the best environment reachable from the joint plans in n. The best environment is theoretically the one where the rest of agents play in favour of α. Which combination of actions is the best to α may also be difficult to estimate. $n.max \geq max\{f(p(\epsilon)) \mid p \in plans(n)\}$

NB^3 computes an 'expected value' for each node, $n.E$, that estimates the value we would obtain if the partial plan in the path, $p = n.path$, and the reached agreements C_α are executed and no extra actions are done. That is $n.E = f((p \cup C_\alpha)(\epsilon))$. Clearly $n.min \leq n.E \leq n.max$. If the computation has to stop before finishing the exploration, NB^3 choses the node with the highest expected value as the solution. Also, the best current node is the one with the highest expected value. NB^3 chooses $n.E$ instead of $n.max$ to avoid selecting nodes containing feasible (although absurd) proposals that would be highly beneficial to α but most probably rejected by the other agents (e.g. selling a pen for a million dollars).

3.4 Pruning

When an offer that α has issued is accepted by agent β then it means that agent β is making a commitment to act upon the environment in a particular way, that

[4]humans may be involved in the MAS.

is, it will perform certain actions from the set of available actions $O \subseteq fea(C_\beta, \epsilon)$. All actions that are incompatible with those in O are unfeasible, therefore we can prune all nodes such that in their paths to the root there is any of the incompatible actions. When an offer made by α to β is rejected the nodes that assumed the actions of β in the offer are also pruned.

4 Algorithm

In this section I give the details of NB^3.[5] The basic data structure is $Node(A)$, parametric on the set of agents A, contains the set of feasible operations for the agents that are not yet explored, *ops*, the explored ones, i.e. the path to the root, *path*, the bounds and expected value, min, max, E, a link to the father, *father* and the label on it, *label*. The initial environment is ϵ_0.

```
Node(A)
  $ops, path : 2^{\mathcal{O}}$;
  $min, max, E : [0,1]$;
  $father : Node(A) \mid nil$
  $label : \mathcal{O} \mid nil$
  $\epsilon_0 : \mathcal{E}$
  ----
  $\forall\, o \in ops.comp(\{o\} \cup path, \epsilon)$
```

The predicate $comp(\{o\} \cup path, \epsilon)$ indicates that any feasible operation o must be compatible with the actions on the path to the root. I assume that the actions $o \in \mathcal{O}$ include the identifier of the agent performing the actions and thus the function $pr_i(O)$ computes the actions performed by agent i in an arbitrary set $O \subseteq \mathcal{O}$.

The following functions are domain dependent and need to be defined for each use case: *fea* determines the set of feasible actions in an environment given a set of actions already committed to and *comp* determines if a set of actions can be performed at the same time in a given environment.[6]

$$fea : 2^{\mathcal{O}} \times \mathcal{E} \Rightarrow 2^{\mathcal{O}}$$
$$comp : 2^{\mathcal{O}} \times \mathcal{E} \Rightarrow \mathbb{B}$$

The following functions are agent dependent:

$$generate : Node(A) \times A \times 2^{\mathcal{O}} \times 2^{\mathcal{O}} \times 2^{2^{\mathcal{O}}} \to 2^{\mathcal{O}}$$
$$h : Node(A) \times 2^{\mathcal{O}} \times 2^{\mathcal{O}} \times 2^{2^{\mathcal{O}}} \Rightarrow 2^{\mathcal{O}}$$
$$f : \mathcal{E} \to [0,1]$$
$$boundmin : Node(A) \times 2^{\mathcal{O}} \times 2^{\mathcal{O}} \to [0,1]$$
$$boundmax : Node(A) \times 2^{\mathcal{O}} \times 2^{\mathcal{O}} \to [0,1]$$

[5] I use Z [9] and classical algorithmic notation.

[6] For instance, in any reasonable environment ϵ, moving box X from position A to position B is incompatible with moving it to position C, that is, $comp(\{move(X, A, B), move(X, A, C)\}, \epsilon) = false$.

where *generate* determines a deal to be sent to another agent when expanding a node given the actions *committed*, *unfeasible*, and *under negotiation*, h is the heuristic that gives as result the set of operations to use to split a node, f is the evaluation function of the agent and *boundmin* and *boundmax* are the bounding functions. These functions are key as the h function determines what part of the space will be explored and in which order, and *generate* has to build attractive proposals that may be accepted and thus prune the tree. Both together have to avoid absurd proposals that would be rejected by the others and slow down the search process.

Algorithm 1 is run by each agent. The main variables are C representing the achieved commitments, U representing the unfeasible actions, i.e. explicitly rejected or withdrawn, and P representing the proposals under negotiation. τ represents the minimum exploration time to wait for a good proposal to be accepted.

Algorithm 1 NB^3

```
Require: $A = \{\alpha, \beta, \ldots, \omega\}$ the set of agents
Require: $\alpha \in A$ the agent performing the computation
Require: $\epsilon_0$ the initial environment.
Require: $t_{max}$ time limit
Require: $\tau$ acceptance time threshold
Ensure: $O \subseteq pr_\alpha(fea(\{\}, \epsilon_0))$ a set of actions to perform
 1: $C, U = \{\} : 2^{\mathcal{O}};\ P = \{\} : 2^{2^{\mathcal{O}}}$
 2: $root : Node(A)$
 3: $Open : \text{seq}\, Node(A)$
 4: $root = create(Node(A))$
 5: $root.ops = fea(\{\}, \epsilon_0);\ root.path = \{\}$
 6: $root.min = 0;\ root.max = \infty;\ root.E = f(\epsilon_0)$
 7: $root.father = nil;\ root.label = nil$
 8: $Open = \langle root \rangle$
 9: $BestMin = root.min$
10: spawn Search'n'speak
11: spawn listen
12: repeat
13:   nil
14: until $t_{now} \geq t_{max} \vee Open = \langle\rangle$
15: kill all threads
16: return $pr_\alpha(Best.path \cup C)$
```

Algorithm 2 is a loop that expands the search tree. The first node in *Open* is selected as the *Best* current node. The algorithm re-computes the set of feasible operations and the bounds, as new commitments may have been added to C that may modify them. Then the function h selects (line 22) the set of operations that will split the node into a set of children. The bounds of the children are computed and then the set of *Open* nodes is sorted. The final loop (lines 35–37) prunes nodes.

Algorithm 2 *Search′n′speak*

```
1: while $Open \neq \langle\rangle$ do
2:   $Best = first(Open)$
3:   $Open = rest(Open)$
4:   for $d \in P \mid time(d) \geq \tau \wedge proposer(d) = x \wedge x \neq \alpha \wedge comp(Best.path \cup C \cup d, \epsilon_0)$ do
5:     $accept(\alpha, x, d)$
6:     $C = C \cup d$
7:   end for
8:   $Best.ops = Best.ops \setminus U$
9:   $Best.min = boundmin(Best, C, U)$
10:  $Best.max = boundmax(Best, C, U)$
11:  $Best.E = f((pr_\alpha(Best.path \cup C))(\epsilon_0))$
12:  for $o \in Best.path$ do
13:    if $o \in fea(x, \epsilon_0) \wedge x \neq \alpha \wedge (\neg \exists\, d \in P. o \in d)$ then
14:      $NewDeal = generate(Best, x, C, U, P))$
15:      for $d \in P \mid \neg comp(d \cup NewDeal, \epsilon_0)$ do
16:        if $d \in pr_\alpha(P)$ then
17:          $withdraw(\alpha, x, d)$
18:        else
19:          $reject(\alpha, x, d)$
20:        end if
21:      end for
22:      $offer(\alpha, x, NewDeal)$
23:      $P = P \cup \{NewDeal\}$
24:    end if
25:  end for
26:  $O = h(Best, C, U, P)$
27:  for $o \in O$ do
28:    $n = create(Node(A))$
29:    $n.ops = Best.Ops \setminus \{o\}$
30:    $n.path = Best.path \cup \{o\}$
31:    $n.father = Best;\ n.label = o$
32:    $n.min = boundmin(n, C, U)$
33:    $n.max = boundmax(n, C, U)$
34:    $n.E = f((n.path \cup C)(\epsilon_o))$
35:    $Open = Open \bullet n$
36:    $BestMin = max(BestMin, n.min)$
37:  end for
38:  $Open = sort(Open, E, \geq)$
39:  while $Open \neq \langle\rangle \wedge ((first(Open)).max < BestMin \vee \neg comp((first(Open)).path \cup C, \epsilon_0))$ do
40:    $Open = rest(Open)$
41:  end while
42: end while
```

Algorithm 3 deals with the house keeping of commitments and unfeasible actions according to the messages received.

Algorithm 3 *Listen*

```
loop
  if $\mathit{offer}(x, \alpha, d)$ then
    $P = P \cup \{d\}$
    $U = U \setminus pr_x(d)$
  end if
  if $\mathit{accept}(x, \alpha, d)$ then
    $C = C \cup d$
    $P = P \setminus \{d\}$
    for $o \in \mathcal{O} \mid \neg \mathit{comp}(\{o\} \cup C, \epsilon_0)$ do
      $U = U \cup \{o\}$
    end for
  end if
  if $\mathit{reject}(x, \alpha, d)$ or $\mathit{withdraw}(x, \alpha, d)$ then
    $U = U \setminus pr_x(d)$
    $P = P \setminus \{d\}$
  end if
end loop
```

In the worst case the tree explores every state and is the complexity is the size of the state space. If the function *boundmin* is exact then the complexity is linear on the depth of the tree [7]. These extreme cases are however very rare. The exchange of offers makes that the heuristic h is informed of the interest of the opponents and thus explores the parts of the space that contain the most plausible deals that would, by being agreed upon, induce a large pruning of the tree and thus provide good average search complexity. A detailed analysis of the complexity is beyond the scope of this paper.

5 Guidelines for the heuristic h

The function h ranking the split operations has to try and guess the most promising ones given the goals of the agent. A good general rule is to start from those that are 'easier', that is, those that α can do without much conflict with the others, then go for the actions involving 'friends', and then with the rest. Also, when a proposal is received it is better to first explore our part of the deal to make sure the deal is interesting to us.

When agent β withdraws a previous offer or α withdraws an offer made previously to β the probability of reaching agreements with β decreases and thus the splitting on actions that might be interesting to β should be delayed. Similarly, when agent β sends an offer to α, or α sends β an offer, the actions that might be interesting to β should be explored as soon as possible by α.

There is always the possibility that agent β commits to an action that β will not execute, intentionally or not. Measures of trust are useful as to determine whether to prune a node or not. That is, if the trust on β is too low and its actions are in the set of commitments we may keep as open some nodes that otherwise could have been pruned to allow for more exploration of plans not involving β. This indicates that the more trustworthy an agent is the more pruning its commitments will cause and therefore the earlier in the negotiation with it the better for the overall efficiency of the search. Trust is therefore one of the important factors in the heuristic that orders the operators to split.

6 Conclusions and further work

In this paper a new negotiation algorithm has been introduced. The motivating use cases are characterised by a huge space of solutions and the particular aspect that co-operation improves the individual outcome. The algorithm uses the offers exchanged between agents to direct the exploration of the space. In this way the search is focused on those parts where agreements are more easily found. The algorithm has been described in detail. As future work I plan to have a probabilistic modelling of the commitments and unfeasible actions to better model agent preferences, trust and reputation. The algorithm is currently being applied to two of the use cases (time tabling and Diplomacy).

References

[1] B. Gendron and T. G. Crainic. Parallel branch-and-bound algorithms: Survey and synthesis. *Operations Research*, (42):1042–1066, 1994.

[2] Nick Jennings, Peyman Faratin, Alessandro Lomuscio, Simon Parsons, Carles Sierra, and Mike Wooldridge. Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[3] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[4] Raz Lin and Sarit Kraus. Can automated agents proficiently negotiate with humans? *Commun. ACM*, 53:78–88, January 2010.

[5] *MJC*^{2}. http://www.mjc2.com/transport_logistics _management.htm.

[6] David Norman. http://www.ellought.demon.co.uk/ dipai/.

[7] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[8] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. The MIT Press, Cambridge, USA, 1994.

[9] M. Spivey. *The Z Notation (second edition)*. 1992.

[10] RJ Willemen. *School timetable construction : algorithms and complexity*. Technische Universiteit Eindhoven, 2002.

[11] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10:673–685, 1998.