

International Transactions on Systems Science and Applications Volume 1 Number 4 (2006) 357-362



A General Framework for Multi-Agent Search with Individual and Global Goals: Stakeholder Search

Alan Fedoruk Jörg Denzinger

Department of Computer Science, University of Calgary 2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4 Email: {fedoruk; denzinge}@cpsc.ucalgary.ca

Abstract: Stakeholder search is a general framework for an extension to the improving on the competition approach paradigm for cooperative search that allows for additional individual goals of the search agents. This framework defines a whole spectrum of possibilities for search systems. Based on a scheme defining interactions, search agents use a given strategy (ranging from cooperative to competitive) to find good solutions for the global search goal that are also good for their individual goals. A stakeholder search system was created to solve instances of the package delivery problem. Experiments with search agents using either a cooperative, a competitive or a stakeholder strategy, between the two extremes, showed that the stakeholder strategy was effective at finding solutions which satisfied both the global goal and many of the individual agent's goals.

Keywords: distributed search, semi-cooperative search.

1. Introduction

Large projects, such as planning the management of public resources or building a new airport involve many individuals, organizations and interest groups—they all have a stake in the project. The lead organization for the project and the interested stakeholders all have goals, which may complement each other, but may also be conflicting or unrelated. For example, the lead organization may want to minimize cost for the new airport while a local residents' group may want to minimize noise levels.

One approach to such a project is to gather all of the goals and centrally attempt to find a plan that satisfies, or comes as close as possible to satisfying all of the competing goals, usually using some form of multi-objective optimization. For a large and complex project, this is a difficult task requiring disclosure and a prioritizing of the goals, which may not be possible as stakeholders may want to keep their goals private to protect trade secrets, or to maintain strategic advantage. Such a process requires considerable analytic resources, many interactions between the parties and a lot of time; and in the end it may still not produce a solution that the stakeholders are all happy with.

The stakeholder approach is for the lead organization to off-load the analysis to the stakeholders and allow them to propose solutions—that they will be happy with—then allow for improvements to these solutions through interaction, and finally choose a solution that best meets the lead's goals. In return for doing this work, stakeholders have the opportunity to influence the outcome of the process as it is one of their solutions that will be chosen; also, stakeholders' goals and methods can now be kept private. Using this approach, the lead organization may not get a solution that meets its goals as well as using the previous approach, but by keeping the final say over the selection, a good solution will still be found. Additionally, the solution will be found using fewer of the lead's resources, and stakeholders may be happier with the outcome. For example, if the residents' group finds two low cost solutions, the solution with the lowest noise levels will be put forward.

Looking at what areas in AI and multi-agent systems can be used to solve problems of this type, networks of computers and cooperative search concepts immediately come to mind. A collection of search agents, representing the lead agency and the stakeholders, work cooperatively to find good solutions; however most cooperative search concepts assume a total cooperation between the agents or computers with a selfless flow of information and agents acting solely for the greater good to find the best solution to the search problem. In the case of large multi-stakeholder projects, this is not a realistic assumption; in addition to the lead's overall goals for the project, each of the stakeholders has individual goals, which they may not wish to reveal. Traditionally this type of competition has been dealt with by market mechanism approaches, like auctions, but these approaches are aimed more at guaranteeing certain minimal quality results. Only very recently have we seen first approaches that mix cooperative search with individual goals of the agents, most often for distributed constraint satisfaction problems solved by dividing the problem instance into subproblems and keeping constraints private or as private as possible (see [5][6]).

This paper presents stakeholder search, a general framework for cooperative search with one global and multiple individual agent goals. Stakeholder search uses the general cooperation paradigm of improving on the competition approach (see [3]), but it is expected that similar frameworks for other paradigms can also be constructed. The framework covers a whole spectrum of possibilities, with stakeholder agents implementing strategies from wholly cooperative, where agents ignore their individual goals and search for solutions which only satisfy the global goal, to competitive, where global goals are disregarded, or a strategy between these extremes.

The stakeholder search framework identifies the key decisions that need to be made to set up an instance of the

search, at both the search level and at the stakeholder level, to achieve the desired results. It is assumed final selection of a solution is made based on the global goal, and that either the solution time available is limited, or the problem is so large that a globally optimal solution cannot be found within practical limits.

The framework was used to create an instance of stakeholder search to solve package delivery problems (PDP): a group of delivery agents must work together to plan a set of deliveries. Experiments with various strategies show that an appropriate middle strategy, between cooperative and competitive, achieves results that are near, when evaluated against the global goal, to those achieved via pure cooperation, but that better satisfy the local goals.

2. From Optimization to Search

A discrete optimization problem, finding an assignment to a set of variables such that a certain goal is achieved, is defined as follows.

Definition 1. Discrete Optimization Problem. *Opt* =(X, D, *Cond*, F) is a discrete optimization problem if:

- $X = \{X_1, ..., X_n\}$ is a set of n variables;
- D = {D₁, ..., D_n} is a set of n discrete, finite domains such that D_i is the domain of variable X_i;
- Cond = { C_1 , ..., C_m } is a set of predicate conditions that solutions must fulfill, $C_i: \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \rightarrow \{true, false\};$
- \mathcal{F} is a goal function, $\mathcal{F}: \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathbb{R}$.

The set of possible solutions *Sol* is defined as the set of all assignments to *X* such that all conditions *Cond* are fulfilled.

Sol = { d_1 , ..., d_n | $X_i = d_i \in \mathcal{D}_i$, and C_i (d_1 , ..., d_n)= true for $1 \le i \le m$ } The optimal solution is $s \in Sol$ where $\mathcal{F}(s)$ is minimized.

For a given X, D, and *Cond*, many different choices for the goal function, F, are of interest; *Sol* stays the same and only the optimal solution, *s*, differs for the different functions. It is also possible to combine several goal functions into a single function and thus find a solution satisfying multiple goals.

Discrete optimization problems are generally solved by search. Using definitions from [3], a search consists of a series of transitions through a set of possible search states. The search begins at a specified initial state; a search control selects the next transition as a function of the current state, that transition is applied and this process continues until a given end condition is met.

Definition 2. Search Process. A triple $\mathcal{P} = (\mathcal{A}, \mathcal{K}, I)$ defines a search process if:

- A =(S, T) defines a search model where S is a set of possible search states and T is a set of possible transitions between those states, T⊂ S×S;
- *K* is a function that defines a search control *K*; *S*→*T*, which selects a transition given the current state of the search;
- I =(s₀, G) defines a search instance where s₀∈S is an initial state for the search and G is a function, G: S→(yes, no), that returns yes when a goal state has been reached, no otherwise.

In the case of a discrete optimization problem, the search moves through a set of search states that reflect *Sol*, the set of possible solutions, directed by the search control. A genetic algorithm's search states, for example, are subsets of the power set of *Sol.* For a branch-and-bound tree search states are trees that construct the elements of *Sol.* The goal function is incorporated into the search control, \mathcal{K} , in such a manner that the control directs the search toward solutions that maximize the goal function. Changes in goal functions only require changes to \mathcal{K} ,

Searches are distributed to improve efficiency, efficacy and to allow parts of the search process to remain private. There are three paradigms for distributed search: improvements on the competition approach, partitioning of the problem into sub-problems and using a shared search state [3]. Improvements on the competition approach have a set of search processes each solving the search problem; the result of the search is the best solution found collectively. Partitioning the problem is the common divide-and-conquer approach: different search processes work on different parts of the problem and the result of the search is synthesized from the partial results. A shared search state approach has a collection of search processes updating a common search state. In all three paradigms, improvements can be made to the search by allowing search processes to exchange information during the course of the search.

A distributed search system is composed of search agents, control agents, and a communication structure to allow information to be exchanged between the agents. A general distributed search system is defined as follows.

Definition 3. Distributed Search System. $DSS = (AG_{Search}, Ag_{Start}, Ag_{End}, Com)$ defines a distributed search if:

- *AGSearch* is a set of search agents, see Definition 4;
- *Ag_{Start}* is a search start agent, which provides a search instance for each of the search agents;
- Ag_{End} is a search end agent, which takes the results of the searches performed by the search agents and creates a final result for the search;
- Com =(Com₁, ..., Com_l) defines a communication structure, an *l*-tuple of sets of data objects, the value of Com is an *l*-tuple val =(c₁, ..., c_l), c_i ∈ Com_i and there is a function, dat : Com ×(1, ..., l)→ Com_l) · · · · ○Com_l), dat(val, i)= c_i

Search agents, consisting of a search process, a shared communication structure and a message function are defined as follows.

Definition 4. Search Agent. A tuple Ag = (P, Com, M) defines a search agent if:

- *P* is a search process, as in Definition 2;
- *Com* is a communication structure, as in Definition 3.
- \mathcal{M} is a messaging function $\mathcal{M}: Com \times S \rightarrow Com$.

In a distributed search system based on improving on the competition approach, the start agent, \mathcal{A}_{gStart} , passes the original search problem to all of the search agents. The agents perform the search and deliver their solutions to the end agent, \mathcal{A}_{gEnd} , which selects one of those solutions as the result of the search. Cooperation between the search agents is realized by the exchange of information, positive or negative, which can be either integrated into the search state or integrated into the search control, or both. For example, an agent may inform other agents of a good candidate solution that may assist them in their search, or an agent may inform other agents of a poor solution that they should avoid.

3. Stakeholder Search

The stakeholder search framework is based on the metaphor of a stakeholder process, used to find solutions to problems where multiple parties have an interest or stake in the outcome. The process is carried out by a stakeholder group minimally consisting of a chairperson and several stakeholders. The lead agency chairs the group and is responsible for managing the stakeholder process and seeing that the lead's goals are met. Stakeholders represent various interests in the problem—they have their own goals—and are responsible for seeing that those interests are met as well as possible within the process and goals set out by the chairperson.

Stakeholder groups are typically given large problems, with a large number of possible solutions and many competing goals. In addition, the group is often under time and other resource constraints. Under these conditions, finding an optimal solution is unlikely and there may be a large number of sub-optimal solutions to choose from. The group strives to find one of these sub-optimal solutions that is closest to the optimum and also meets as many of the stakeholders' goals as possible. Stakeholders are doing the bulk of the work; they buy-in to the process because it gives them a say in the final outcome. Stakeholder groups are both cooperative and competitive: the stakeholder group members cooperate so that a good solution is found, one which also meets their individual goals, but those goals may be in conflict so competition is needed as well.

A stakeholder process consists of a series of meetings. At the initial meeting, the problem, the chair's (lead's) goal, and the rules of the process are communicated to the stakeholders. At each subsequent meeting discussion takes place, mainly involving exchanges of possible solutions to the problem; the solutions presented depend on what strategy the stakeholders employ. Between meetings the stakeholders further analyze the problem– incorporating information gleaned from the meetings– seeking solutions that meet their own and the lead's goals. At the final meeting, a solution is chosen based on the lead's goal; the solution will also represent the stakeholders' goals.

Though multi-objective optimization (MOP) techniques solve problems similar those solved by stakeholder search, stakeholder search differs in several ways. MOP techniques require stakeholders to reveal all goals, while stakeholder search allows stakeholder goals to be kept private. In MOP goals are ranked or weighted, depending on which MOP is being used, which may not be possible in the stakeholder setting. Some MOP techniques are designed for at most two objective (goal) functions, or are only practical with a small number of objective functions; stakeholder is designed to work with large numbers of stakeholders and goals. Finally stakeholder solutions are selected on the basis of the lead's goal function, so in effect the process is single objective optimization; the stakeholder's goals are realized as a by-product of the process.

Stakeholder search is an extension of the improvement on the competition approach paradigm: search agents work on a problem independently, and a result is selected from the solutions found. A stakeholder search system consists of a chair agent, a collection of stakeholder agents, and information required to carry out the search. Building on Definition 3, a Stakeholder Search System is formally defined as follows. **Definition 5**. Stakeholder Search System. A tuple $SSS = (Ag_{Chair}, AG_{SH}, Com, Prob, Scheme, F_{Global})$ defines a stakeholder search system if:

- $Ag_{Chair} = (Com, \mathcal{M}_{Chair})$ is a Stakeholder Chair Agent;
- $\mathcal{A}G_{SH} = \{\mathcal{A}g_{SH_{1}}, ..., \mathcal{A}g_{SH_{n}}\}$ is a set of $n \ge 1$, Stakeholder Agents;
- *Scheme* defines an interaction scheme and interaction constraints;
- *Prob* is an instance of an optimization problem from Definition 1;
- \mathcal{F}_{Global} is a goal function, as defined in Definition 1;
- *Com* is a communication structure, as defined in Definition 3.

The Stakeholder Chair Agent fills the role of both the start and end agents, \mathcal{Ag}_{Start} , \mathcal{Ag}_{End} from Definition 3. To begin the search, it distributes the search problem, *Prob*, the *Scheme* and the global goal, \mathcal{F}_{Globab} to the stakeholder agents.

The search proceeds as a series of rounds, each round consisting of a search phase and an interaction phase. The length of each round and the structure of the interaction is defined by the *Scheme*.

Definition 6. *Scheme*. A tuple *Scheme* = (*Interact, Constr*) defines interactions in the search where:

- *Interact* defines the structure of the search, the number and length of rounds, the duration of the search and interaction phases and the structure of the interaction—what types and quantity of information is to be exchanged, and what dialogues will take place.
- *Constr* is a set of constraints that all stakeholder agents have to fulfill. For example, during an interaction phase, only valid solutions might be permitted to be exchanged, or only solutions better than the current known best solution can be exchanged.

Stakeholder agents perform the search, and have a search process to do so, they communicate with the chair agent and other stakeholder agents via the shared communication structure and their message function. Formally, a stakeholder agent is defined as follows.

Definition 7. Stakeholder Agent. A tuple $Ag_{SH} = (P, M, Com, F_{Local, Strategy})$ defines a Stakeholder Agent where:

- *P* is a search process as given in Definition 2;
- *M* : *Com* ×*S*→*Com* is a Stakeholder Message Function with four functions:
 - Get: read data from the communication structure,
 - *Put*: write data to the communication structure,
 - *Proc*_{Inc}: select and process incoming data,
 - *Proc*_{Out}: select and process outgoing data;
- *Com* is a shared communication structure;
- \mathcal{F}_{Local} is a local goal function on A.
- *Strategy* is the stakeholder agent's strategy, a set of rules which determine the tactics it will use to meet its goals within the confines of *Scheme*.

The stakeholders in a stakeholder group come to the process with a strategy in mind. That is, what tactics will they use, within the confines of the rules of the process, to attempt to best meet their goals. For example, they can act selflessly and work toward the goal of the lead, or they can be selfish and work only toward their individual goals or they may adopt a strategy between these two extremes. Similarly, a stakeholder agent must have a *Strategy*.

A stakeholder agent comes with a local goal function, \mathcal{F}_{Local} , a search model, \mathcal{A} , and a strategy, *Strategy*, and as the first step of the search, the stakeholder agent receives the problem description, *Prob*, the global goal, \mathcal{F}_{Global} , and the scheme, *Scheme*, from the chair agent. This information is used to initialize the stakeholder agent and prepare to search. A search control, $\mathcal{K}_{,}$ is created which reflects the global and local goals and the strategy. Using the problem instance and the scheme a search instance, *I*, is created. The detailed instantiations of the message function are created, which depend on the interaction scheme and constraints laid out in *Scheme* and *Strategy*.

By using different strategies it is possible for a stakeholder search to be instantiated at any point in the spectrum from purely cooperative to totally competitive. Stakeholder agents are free to use any tactics that do not contravene *Scheme*. The *Scheme* that is set out for a search is the counter-balance to the stakeholder tactics. For example, a stakeholder agent may use a tactic of holding back good solutions until later search rounds with the hope of finding the minimum global quality solution that will be the final result, by observing what other stakeholders are presenting. To prevent this type of tactic, a *Scheme* could leave the number of rounds open. Privacy is one of the reasons for distributing search and stakeholder agents are not required to reveal anything more than what *Scheme* requires for interaction.

4. Related Work

The stakeholder search framework extends the work presented in [3], which gives formal definitions for search and distributed search and identifies three paradigms for distributed search: improvements on the competition approach, partitioning of the problem into sub-problems and using a shared search state. Stakeholder search is an improvement on the competition approach, a paradigm commonly used in cooperative searches. The Island Model for genetic algorithms is one example, which has a collection of sub-populations, evolving separately, and communication via migration of individuals [1]. In the TECHS approach, [4], a framework similar to our stakeholder search is developed, but stakeholder search expands on this by allowing agents to also work toward their individual goals.

The distributed constraint satisfaction work done in [5][6][7] are examples of a stakeholder-like approach using a partitioning of the search space paradigm rather than improving on the competition approach. In these works a constraint satisfaction problem is partitioned and the partitions given to search agents to solve. Each search agent is responsible for an individual variable and its constraints. The constraints are kept private with agents only revealing whether proposed variable assignments violate the constraints. These methods have a global goal, a solution to the constraint problem, as well as individual goals, to satisfy the individual constraints, without revealing them too much.

Stakeholder search solves problems similar to multi-objective optimization problems. Multi-objective optimization attempts to find a Pareto optimum solution between the multiple objectives, usually by combining the multiple objective functions into a single function [2]. Stakeholder search differs from MOP as described in Section 3., with the key difference in selecting results; stakeholder only has one objective function, the global goal function, fulfilling the other functions is a by-product of the distributed search.

Conflicts in competitive agent environments are often solved by techniques such as auctions, voting or negotiations. Stakeholder search can be instantiated to mimic or incorporate all of these techniques.

5. The PDP Solver

The Package Delivery Problem (PDP), a variant of the traveling salesman problem (TSP), is used as an example of a discrete optimization problem. In a PDP, a parcel delivery company has a set of packages to deliver and there are a group of delivery persons (drivers) who are available to make the deliveries. The company's goal is to get all the packages delivered at the lowest cost; the drivers have a goal of minimizing the amount of work she does.

This problem was chosen as a test problem for stakeholder search as problem instances are straight-forward to specify, a relatively small instance of the problem (e.g. 20 packages, 3 drivers) produces a large search space with many possible solutions, and there are many possibilities for specifying global and local goals, which are readily translated to goal functions. Example goals include: minimize distance or time traveled for all drivers or for an individual driver.

Building on Definition 1 a PDP is defined as follows.

Definition 8. Package Delivery Problem. PDP = (n, m, Cost, F) is a PDP if:

- *n* is the number of packages to deliver;
- *m* is the number of drivers available to make deliveries;
- Cost is a set of l cost matrices; Cost = {c_i | c_i is an n × n matrix with c_i[j, k]=a cost of moving from the delivery location of package j to that of package k_a 1 ≤ i ≤ l};
- \mathcal{F} is a goal function, $\mathcal{F}: Cost \times \mathcal{D}_i^n \to [0...1]$.

then

- $\mathcal{D}_i = \{(j, k) \mid 1 \le j \le m \text{ and } 1 \le k \le n\} \text{ for all } i, 1 \le i \le n, \text{ for all } \mathcal{D}_i \in \mathcal{D};$
- Cond = $\{X_i \neq X_j \text{ for } i \neq j\}$.

The distributed PDP solver fits the stakeholder search framework and has the following characteristics:

- There can be 1 or more stakeholder agents, each representing a driver.
- The global goal is to minimize the total distance traveled by all drivers.
- The search will proceed for a pre-determined number of rounds. Each round consists of a search phase of consistent and predetermined length.
- During the interaction phase, stakeholder agents will send to the chair agent 1 solution. The chair will then send all solutions collected, to all stakeholders.
- All shared solutions must be valid. All packages must be delivered and deliveries must be consistent between drivers.

Communication in the solver is always between stakeholder agents and the chair; information exchanged is of two types, process information, such as the problem instance or scheme, and solutions. *Com* is structured with 4 data areas for each stakeholder agent: a data area to process information coming from the chair agent, a data area sending process information to the chair agent, and a similar pair of data areas for exchanging solutions with the chair agent.

Problem instances are as given in Definition 8. Problems have a single cost matrix consisting of distances between all package delivery locations. The global goal function sums the distances that all drivers must travel to deliver their assigned packages.

The *Scheme* consists of the number of rounds, the length of the search phase, and a rule that specifies that 1 valid solution is to be delivered by each stakeholder agent during each interaction round. Valid solutions meet the conditions of a solution given in Definition 8.

Stakeholder (driver) agents can use different search models. For the PDP solver, three search models are available: a genetic algorithm (GA) using mutation, crossover and swap as genetic operators; a tabu search (TABU) which will jump to a new area in the search space when a good solution has not been found in a given number of transitions; and a branch-and-bound tree (BANDB). Solutions from other agents are incorporated into the GA by including the incoming solutions in the population. TABU keeps the incoming solutions in a list and uses that list as jump locations. The BANDB uses the incoming solutions the same way that it would use a solution that it found it its own search.

Agents are equipped to implement one of a range of strategies, specified by a triple: *PDPStrategy* = (Add, Search, Select), where

- $Add \in \{true, false\}$ If true, solutions distributed by the chair agent during the interaction phase are incorporated into the local search, otherwise they are ignored.
- Search ∈ {Local, Global, Combo} Direct the search by either the local goal function, the global goal function or a combination of both.
- *Select* ∈ {*Local*, *Global*, *Combo*} Select solutions to submit to the chair during the interaction phase of each round by the local goal, the global goal or a combination of both.

In this instance the *Combo* goal function is the average of the local and global goal function. The strategy chosen is realized by appropriate instantiations of the message function and the search control.

6. Experimental Results

A set of 10 randomly created PDP instances were created for testing. All of the problems have 10 packages and 3 drivers, n = 10 and m = 3. Each problem instance has a single cost matrix consisting of the distances between each of the delivery locations. To allow results to be expressed as a percentage of the optimum solution, un-time-constrained search runs were performed to find the optimal global solution for each test problem.

The goal of these experiments is to evaluate a cooperative, a competitive, or a stakeholder strategy, between the two, by seeing how well solutions found by the PDP Solver meet the global and local goals. Results are not compared to other systems since we are trying to see how close to the optimum, which is known, various strategies will bring us. There is no other work to compare this to.

It is expected that fully cooperative strategies will yield good global solutions, that competitive strategies will yield poor global solutions and that a stakeholder strategy will yield solutions that are good when evaluated by both of the local and global goal functions.

Additional testing was performed to find three strategies that represent a competitive, cooperative, and a stakeholder strategy, respectively. These are:

- $PDPStrategy = (f alse, Local, Local) \rightarrow Competitive.$
- $PDPStrategy = (true, Global, Global) \rightarrow Cooperative.$
- $PDPStrategy = (true, Combo, Combo) \rightarrow Stakeholder.$

The search *Scheme* specifies a search consisting of 6 rounds, each with a search phase lasting 60 seconds. Preliminary experiments showed that, for this set of problems, 6 60-second rounds are enough to find a good solution and get some benefit from the interaction between stakeholder agents, but are still constraining enough that the process does not simply find the optimum.

A test consists of running the PDP solver against a particular problem instance for one of the three strategies (cooperative, competitive, stakeholder). All tests used three stakeholder agents, using a GA, TABU and BANDB search respectively, with all three agents using the same *PDPStrategy*. Each agent was run on its own computer.

A trial consists of a series of tests of all problem instances using all strategies. Table 1 shows the results of ten trials, with the results averaged to mitigate any random effects caused by OS scheduling or network traffic. Each row of the table shows the results for the named problem; the first three columns show, for each of the strategies, how closely the solution met the global goals and the final three columns show how closely the solution met the local goals. Results are given as percentages of the optimum. For the global goal the optimum for each problem was determined prior to the trials. For the local goal, stakeholder agents always have an optimum distance traveled of 0, that occurs when all deliveries are assigned to other drivers. To obtain a single number for the local goals, the individual results were averaged.

Table 1. Experimental results as percentage of optimum for 10 trials.

Problem	Global % Opt.			Local % Opt.		
	Comp	Coop	Stake	Comp	Coop	Stake
Test10-3A	41.99	90.34	92.44	24.67	38.21	35.05
Test10-3B	41.84	83.19	80.37	27.21	41.05	34.55
Test10-3C	43.37	86.21	90.30	27.47	41.40	50.94
Test10-3D	52.58	90.14	92.44	7.57	21.77	34.46
Test10-3E	71.07	93.26	92.62	34.75	28.64	38.11
Test10-3F	51.80	91.14	91.38	34.33	67.09	67.09
Test10-3G	42.95	91.96	86.96	18.07	37.82	35.00
Test10-3H	46.86	86.97	85.21	21.60	39.08	43.60
Test10-3I	49.90	86.41	89.14	29.63	31.01	36.80
Test10-3J	52.60	78.53	90.26	34.12	66.91	66.95

Table 1 shows that the competitive strategy performs worse than cooperative and stakeholder for the global goal in all cases, and in all but one case for the local goals. Using a competitive strategy, stakeholders search for solutions that only satisfy their individual goals, but the result of the search is chosen based on the global goal, a "contradiction", which gives the poor results when evaluated against the global goal. These results show the average of the three agents for the local goals; one of the agents will have done well, but overall the local goals were not met.

The cooperative and stakeholder approaches give similar results for the global goal. Three problems yielded results very nearly the same, 5 with the stakeholder approach ahead and 2 with cooperative ahead. When using the cooperative approach all the agents are using the global goal and-due to the cooperation-will converge toward similar regions of the search space. However, in the stakeholder approach, the stakeholder agents are using a goal function which combines the global and individual goals; this spreads the agents out within the search space, giving more chance of finding the better global solutions observed. This can also account for the results observed for the local goals; the stakeholder strategy gave results the same as or better than a cooperative strategy for 8 of the 10 problems.

7. Conclusion and Future Research

There are four main benefits that come with the stakeholder search framework: It allows for a whole spectrum of strategies from cooperative to competitive; It allows great flexibility in the instantiation of systems opening large areas for further research; It allows processing to be off-loaded; It preserves privacy of goals, methods and strategy.

Experimental results show that a cooperative strategy produces solutions that meet the global goals very well. A stakeholder, between cooperative and competitive, strategy also produces solutions that meet the global goals very well, in some cases better, and meet the local goals better, in 70% of the test cases, than the solutions found using a cooperative strategy.

Since the stakeholder framework defines such a wide range of possible instantiations, there are many possibilities for further research. The effect of various combinations of *Strategy* and *Scheme*, needs to be investigated to find instantiations of interest, and to place the instantiation within the spectrum of the framework.

The interaction phase of the PDP solver is a very simple

exchange of single possible solutions. This can be made more sophisticated and the impact on search results measured. This opens up a whole area of complex interaction types such as negotiation and argumentation, and allows agents to employ agent-modeling techniques to improve strategy and tactics.

References

- E Alba and J M Troya, A survey of parallel distributed genetic algorithms. Complexity, Vol. 4, No. 4, 1999, pp. 31-52.
- [2] C A Coello, An updated survey of GA-based multiobjective optimization techniques. ACM Computing Surveys, Vol. 32, No. 2, 2000, pp. 109-143.
- [3] J Denzinger, Conflict handling in collaborative search. In C Tessier, L Chaudron and H Müller, eds., Conflicting Agents: Conflict management in multi-agent systems, Kluwer Academic Publishers, 2000, pp. 251-278.
- J Denzinger and T Offermann, On cooperation between evolutionary algorithms and other search paradigms. Proceedings of 1999 Congress on Evolutionary Computation (CEC'99), Washington, 1999, pp. 2317-2324.
- [5] P J Modi, W-M Shen, M Tambe and M Yokoo, An asynchronous complete method for distributed constraint optimization. In Proceedings of Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), 2003, pp. 161-168.
- [6] M C Silaghi and B Faltings, A comparison of distributed constraint satisfaction techniques with respect to privacy. In Proceedings of Third International Workshop on Distributed Constraint Reasoning, at International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02) Bologna, Italy, 2002.
- [7] R Zivan and A Meisels, Concurrent backtrack search on discsps. Proceedings of 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS'04), 2004, pp. 776-781.