“GenPlan”: Combining Genetic Programming and Planning

C Henrik Westerberg John Levine

15th September 2000

Division of Informatics,
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
carlw@dai.ed.ac.uk johnl@aiai.ed.ac.uk

Abstract. Planning is a difficult and fundamental problem of AI. An alternative solution to planning may lie in applying Genetic Programming to the planning problem. As such a Genetic Planner was constructed to assess the feasibility of this idea. This paper introduces the topics of Genetic Programming and Genetic Planning and introduces the algorithm used to implement the Genetic Planner. The Genetic Planner was applied to three classical planning domains: STRIPS Blocks Domain, Briefcase Domain, and the Logistics Domain. The Genetic Planner produced good results for both the STRIPS Blocks Domain and the Briefcase Domain. However further work is required before it can solve any problem from the Logistics Domain besides the trivial ones. There is also some comparison of GenPlan with both BlackBox and SINERGY. The first implementation provided many avenues for further research: quick partial plan formation for seeding the Genetic Planner's initial population, more intelligent fitness functions, and an intelligent form of crossover and mutation. Further research into the feasibility of the Genetic Planner to plan in alternative domains besides classical planning is also important.

Introduction

This paper presents an investigation into the feasibility of using Genetic Programming (GP) to solve classical planning problems. A Genetic Planning system was constructed based on information from [1] and [4]. Experiments were done on three different planning domains to assess the ability of the Genetic Planner. The Genetic Planner is also compared to one other Genetic Planner called SINERGY where possible [5] [6] [7]. There is also some comparison with the stochastic planner BlackBox [3].

Motivation for the main idea comes from supplying a new avenue for attacking the difficult problem of planning. GP is a general purpose problem solving technique that has had some considerable success in other areas of AI and it is hoped some of this success can be translated to planning. The possibility for applying GP to planning has already been demonstrated to work by [9] and [5]. However the evidence supplied so far is still piecemeal compared to the abundance of work done on the planning problem using traditional or current methods. The GP algorithm itself is highly xparalisable and the algorithm also runs in constant memory unlike other search strategies. This is important as it is often the case that the search spaces involved are huge and require very large amounts of memory. Finding a solution is difficult in the planning domain, but the solution is not the last step as one is most often interested in an optimal solution. Genetic

Algorithms, which Genetic Programming is based on, has often been used as an optimising tool. GP may offer ways to find find plans with different characteristics such as robustness or expense by tailoring the fitness function to assign more fitness to individuals that have these desirable characteristics. This is a new area of research opening up for planning and promises much excitement and interest.

The Genetic Programming approach to planning is very different to the traditional approach of solving the planning problem. The first work on planning is termed classical planning and is characterised by the planner reasoning logically about possible actions and how the actions relate in order to arrive at a correct plan. The Genetic Planner works by having a population of random plans with each member assigned a fitness via a fitness function. The Genetic Planner proceeds by selecting the most fit plans from the population and then recombining them to produce new individuals for the next population. This new individuals are hopefully more fit than the previous population. In this way the population is evolved to the correct solution.

The system had some very positive results. For instance in the Blocks World Domain the Genetic Planner can solve large problem instances such as large-c and large-d from the BlackBox distrubition [13]. Large-c is a problem that contains 15 blocks and large-d contains 19 blocks. The Genetic Planner could also find solutions to smaller problem instances. The same is true for the Briefcase Domain. The Genetic Planner was able to solve the same problems as SINERGY but also using less generations to do so. The Logistics Domain showed that the simplest representation and fitness function are not good enough to overcome all problems. But this is an important result as it highlights areas for further development.

The paper will proceed to discuss the current work on Genetic Planning and how it differs to the work presented in this paper. This is followed by a quick introduction to Genetic Programming and Genetic Planning. The next section covers the experimental results from the three planning domains the Genetic Planner tried to tackle. Finally there will be some conclusions on the project and some discussion on new directions to take the project in the future.

Current Work

The originality and need for work on this idea is reflected by the amount of material available. There were two papers that were closest to idea of a Genetic Planner. The first was “Genetic Programming and AI Planning Systems” [9]. The second was “A general-purpose AI Planning System Based on the Genetic Programming Paradigm” [5]. There also two other papers written by Muslea which describe the same planner [6] [7] .

Lee Spector managed to achieve three goals with his Genetic Planner. The first was that his planner was able to produce a maximally fit plan for the Sussman Anomaly. The Sussman Anomaly was an important problem with first generation planners, as they often didn’t find the optimal solution first, due to the interaction between goals. The second goal achieved was to find a general plan for a specific goal state from a range of initial states. A general plan that worked in this manner was created, and even worked on some additional initial states that it had not seen before. The third goal achieved was to find a plan that would solve a range of goals from a range of initial conditions. The Genetic Planner was able to find just such a general plan for three blocks problems that could solve any initial state to goal state problem. The general plan even solved some four-block problems. Spector relied on the code that is available from John Koza [4]. In our project we were only ever looking for specific plans from specific initial states to specific goal states. However this work is interesting as showed that there might exist general planning rules for specific problems domains that the Genetic Planner might be able to learn. More importantly these rules may be transferable to other domains. These rules could be packaged up as functions or shortcuts and be used to significantly shorten planning time.

Ion Muslea's Genetic Planner is a planner that works on start state/goal state problems that characterises traditional planners. Muslea did not attempt to find any general plans like Spector's system. Muslea created a Genetic Planner called SINERGY. SINERGY was created in order to compare genetic methods with an older traditional planner called UCPOP [10], which was state of the art in its time. SINERGY was run on several problem domains: Blocks World Domain, Briefcase Domain, and a type of robot navigation and blocks pushing domain. Our work did not consider this robot navigation and blocks pushing domain. In the end SINERGY was able to handle problem instances that were two orders of magnitude larger than the ones solved by UCPOP. These two papers represent the current research with Genetic Planners.

Genetic Programming

Genetic Programming (GP) is a relatively new technique derived from the ideas of Genetic Algorithms (GA). GP is an automatic programming technique that manipulates programs rather than the fixed length chromosomes of GAs. GP uses automatically generated random programs and breeds them using "survival of the fittest" to create new and better programs. The programs are bred in order to produce some desired output based on the input. In this way programs can be "evolved" to solve a wide range of problems.

The primitive elements of GP are terminals and non-terminals. A terminal is a value that can be returned by a function, where as a non-terminal is a function. These non-terminals and terminals then combined to form programs called candidates. In order for GP to work programs must at least be executable without fatal error. One way of achieving this is by satisfying the closure property. The closure property states that "every terminal in the terminal set and every value that may be returned by any function in the function set must be acceptable as an input for every argument position of every function in the function set" [4].

The next component to implement is the fitness function. The fitness function is used to describe how good a particular program is at solving the problem being considered. The fitness function is often implemented by running the program on a number of fitness cases. Each run on a fitness case produces output. The difference between the expected and produced output can be used to assess the fitness of the candidate in question.

The GP algorithm begins by creating a random population of programs This process is called seeding. Each of the programs is executed and assigned a value by the fitness function. The next step is check to see if the some suitable terminating criterion has been reached. If so the algorithm terminates with the best program as output. Otherwise a new population is constructed from the old one. A fitness sensitive selection method, biased towards individuals with higher fitness, is used to select individuals form the population to have genetic operators applied to them. The selected individuals or parents are then used as input to the genetic operators. The genetic operators are used to create new programs for the next population. The two main genetic operators are crossover and reproduction. Crossover involves taking two parents and swapping a number of fragments of each parent between them. Reproduction takes a parent and duplicates it. This result is then added to the next population. The creation of a new population is called a generation. Once the new population has been created the fitness of the individuals is assessed and the termination criterion is checked once more. The termination criterion can be based on a minimum fitness value an individual has to reach or a prescribed maximum number of generations that can't be exceeded.

Genetic Planning

Genetic Planning is therefore the application of GP to Planning. Genetic Planning works on plans as opposed to programs. Programs and plans are analogous as they can both be considered an ordered set of instructions. The same first step applies and that is to produce an initial population of plans this time. The Genetic Planner then evolves the population of plans in order to produce a plan that will achieve some set of goals. This is very different to a traditional planner which works by taking the initial state, declarative operators and goal state as input. The traditional planner then reasons logically about the effect of the operators on the world state and attempts to produce the correct sequence of operators to achieve the goals. The evolution works by applying "natural selection" to the populations. Only those individuals with good fitness are selected and are allowed to breed with other individuals to produce the next population.

The Genetic Planner follows the same algorithm as used in Genetic Programming. First a population of random plans is created. The population is assessed using a fitness function. One possible fitness function is to count the number of goals the plan satisfies after it is finished with the initial state. This proportion can then be assigned to a particular candidate. In order for this to happen the plans must be simulated to determine their effect on the initial state. The plans are simulated by applying the proceduralised versions of actions to the initial state. The procedural versions update the world state if the action can be applied. In this way the plan can then be simulated using the current state as input. The fitness function then uses the results of the simulation to attach a fitness estimate to the plan. Plans are then selected for breeding and a new population is constructed. A simple method to do selection is tournament selection. Tournament selection works by pulling randomly selected individuals out of the population and into a subset. The size of this subset is set by the user. The individual with the highest fitness is then selected to be a parent. This process is done again to get the other parent. This step is required if crossover is the recombination function or if more than one parent is needed. Crossover then combines the two parents to produce offspring for the next population. In GenPlan, crossover is 1-point and only produces a single offspring. Once the new populations is created the termination criterion is checked. Crossover and reproduction are often choosen in proportion. For example 80% of the new population can be made from crossover and the remaining 20% from reproduction.

The generational algorithm is now presented in a clearer form [1]:

1. Initialise the population
2. Evaluate the individual plans in the existing population. Assign a numerical rating or fitness to each individual.
3. Until the new population is fully populated, repeat the following steps:
   (a) Select an individual or individuals in the population using the selection algorithm
   (b) Perform genetic operations on the selected individual or individuals
   (c) Insert the result of the genetic operations into the new population
4. If the termination criterion is fulfilled, then continue. Otherwise, replace the existing population with the new population and repeat steps 2-4.
5. Present the best individual in the population as the output from the algorithm

The generational algorithm used in GenPlan follows this one closely.

Experimental Results

Each of the following experiments used the following experimental set-up unless otherwise indicated.

- Maximum number of generations 50: the number of generations controls how long the Genetic Planner searches for a solution. Nearly all the problems considered are solvable in 50 generations. This is also the figure suggested by Koza [4].
- Population size 1000: the population size controls the number of individuals there are in the population. The larger the population size the more likely the planner is going to solve the problem if at all and require less generations to do so.
- Initial starting plan lengths range between 15 and 30 actions. This is another parameter used to control the amount material the Genetic Planner has to work with. These numbers can be increased or decreased depending on the difficulty of the problem.
- Tournament size of 4. This parameter controls the size tournament used for selection. Larger sizes can be used for faster convergence but run the risk of premature convergence.
- Maximum plan length was 100. This parameter controls how long a plan is allowed to grow in terms of actions. More difficult problems often need a longer length than this.
- Genetic operator proportion, 80% crossover and 20% reproduction. This means that crossover is used 80% of the time to produce new individuals for the next population and reproduction is used the remaining 20% of the time.
- No mutation was used. This is decision based on Koza's decision not to include mutation during his initial case studies [4].
- The fitness function used attributed most of the fitness to those plans which achieved more goals. There was also a small proportion of the fitness value going to amount of working actions in a plan.

This set-up is by no means optimal and was deliberately chosen. For better behaviour on more challenging problems the set-up can be tweaked for the problem in question. This set-up was chosen intentional to demonstrate the general effectiveness of the Genetic Planner.

Blocks World Domain

The Blocks World domain is one of the original problem domains in classical planning. This domain is based on the STRIPS domain. The problem revolves around a robot arm picking up and stacking blocks. The blocks start of in an initial order and the planner must plan the correct sequence of robot actions to achieve the goal state, as prescribed by the porblem. The two actions used in this implementation were newtower(X) and puton(X,Y) [9]. They can be more explictly described as:

```
Operator: newtower(X)
Preconditions: on(X,Y), clear(X)
Add List: on(X,table), clear(Y)
Delete List: on(X,Y)
```

```
Operator: puton(X,Y)
Preconditions: on(X,Z), clear(X), clear(Y)
Add List: on(X,Y), clear(Z)
Delete List: on(X,Z), clear(Y)
```

The Blocks World Domain is important because it is one of the benchmarking domains used to compare different planners. It is also important historically as one of the original planning problem domains. Both Muslea and Spector used the Blocks Domain as the basis for some of their experiments. It was hoped that Muslea's SINERGY could be compared to this Genetic Planner, however the author was unable to provide the exact problem specifications which would have allowed this.

The experiment was based around a series of problems, which were selected in increasing order of difficulty. This experiment will determine how well the Genetic Planner copes with increasingly difficult Blocks World problems. There are two ways to increase the difficulty to the problem. The first is to add more blocks to the problem. This increases the "action space" the Genetic Planner has to deal with. The action space is the space of all valid actions that could possibly make up a plan. Actions which contain duplicate operands are not valid. The alternative is to increase the amount of similarity between the goal state and initial state but still requiring a large number of actions to reach the goal state. These problems are difficult to deal with as they give an impression that the planner is close to solving the problem. These problems arise partly from the fact that the representation of the problem does not give a truthful account of the problem and is also partly due to the fitness function used.

The following list of problems were either created by hand for experimentation purposes or taken from the current planning system PRODIGY (problems 4-8) [11]. The following experiment was chosen to demonstrate the effectiveness of the Genetic Planner and to show that not all problems are yet within its grasp. The following results were gained from experimentation with the Genetic Planner using the default parameters. For each Blocks World problem the Genetic Planner was run 10 times with the best and average number of generations recorded.

Instance	Description	Best	Average	Action Space	BlackBox Solvable?
BWP-1	3 blocks, Sussman	0	0	9	yes
BWP-2	4 blocks, Reverse	0	0	16	yes
BWP-3	8 blocks, (1 stack to 2 stacks)	3	5.4	64	yes
BWP-4	7 blocks	2	4.1	49	yes
BWP-5	Large a 9 blocks	4	7.4	81	yes
BWP-6	Large b 11 blocks	12	19.3	121	yes
BWP-7	Large c 15 blocks	X	X	225	no
BWP-8	Large d 19 blocks	X	X	361	no
BWP-7*	Large c 15 blocks	82	194.6	225	no
BWP-8*	Large d 19 blocks	225	434.3	361	no

Using the standard set-up of the Genetic Planner would knowingly produce a result where not all problems were solvable. The problems marked with Xs were unsolvable with this set-up. However it is possible to extend the power the Genetic Planner at the expense of computer power. For example, performance of the genetic algorithm to solve problems improves with increased population size and an addition mutation. BWP-7 and BWP-8 are solvable using a population size of 2000 and a maximum plan length of 200 as indicated in the table with the asterik. This is a good result as Black Box running on a Ultra 4 with 2 Gigabytes of RAM is unable to solve this two particular problems as it runs out of memory. We also tried BlackBox on a Blocks World Domain that closely modelled our own. These new operators, where the pick-up and put-down acition are combined into one operator, did improve performance on large-b but did not help BlackBox to solve large-c and large-d. GenPlan runs in constant memory bounded by the population size and maximum plan length. As expected the Genetic Planner required more generations to solve the increasingly difficult problems.

One of the original goals of the project was to get the Genetic Planner to work on simple problems involving only three or four blocks. These results far exceed those initial expectations and these results reflect the power of this simple but effective way for plan generation. These results are positive indication of the power of genetic planning. They show that genetic planning is feasible and offers an alternative solution to the planning problem.

Briefcase Domain

The Briefcase Domain is a newer planning domain which can also be described as a classical planning problem domain. In this domain there are a number of briefcase objects and locations. Each object needs to be placed in the correct location. This is done by placing the object in a briefcase and then moving the briefcase to another location and removing the object from briefcase.

The briefcase domain is a valuable problem domain as each of the operators has a high level of interaction with other operators as they work on the problem. The implementation of this problem domain will allow comparison between this Genetic Planner and SINERGY [5] as the author was able to provide an exact description of all the problems he devised and experimented with for this domain. For a complete description of the operators please see [6].

The problems Muslea provided make up numbers BP-3 through BP-9. One of the ways to make the problems harder is to increase the number or objects, locations, and briefcases. This in turn increases the action space. We attempted to match the experiment as closely as possible to Muslea's but this is impossible, as explicit implementation details, like which type of selection method was used, are unavailable. The following experiment was done using the same standard experimental set-up, except that the population size was changed to 2000 to reflect Muslea's population size. For each of the Briefcase problems the Genetic Planner was run 10 times. The average number and the best number of generations to the first correct solution was recorded.

Instance	Description	GP Best	GP Average	SINERGY Results	Action Space
BP-1	2 objs, 2 locs, 1 briefcase	0	0	-	6
BP-2	4 objs, 2 locs, 1 briefcase	0	0.2	-	10
BP-3	4 objs, 5 locs, 1 briefcase	2	3	24	13
BP-4	5 objs, 5 locs, 1 briefcase	0	2.6	2	15
BP-5	5 objs, 5 locs, 5 briefcases	3	6.1	15	55
BP-6	10 objs, 10 locs, 1 briefcases	9	12.6	37	30
BP-7	10 objs, 10 locs, 2 briefcases	20	22.7	44	50
BP-8	10 objs, 10 locs, 5 briefcases	30	38*	95	110
BP-9	10 objs, 10 locs, 10 briefcases	52	58.3*	100	210

The two results with an asterisk indicate that the Genetic Planner was not always able to solve the problem within fifty generations. For BP-8 the Genetic Planner solved the problem seven out of ten runs and for BP-9 three out of ten runs. Predictably the Genetic Planner required more generations to solve the increasingly more difficult problems. This is due to the increasing action space for each problem. The large population size had two effects. The first was to make each run very time consuming, approximately 30 minutes to solve BP-8. The second effect was to make each problem solvable by the Genetic Planner. Most importantly, however, the Genetic Planner was able to solve each of problems using less generations then SINERGY [5]. This could be attributed to any number of implementational details, but possibly the most important of which was that a linear genome was used to represent candidates [12].

The results from this domain duplicate those of the Blocks Domain in that the difficulty of problems solved exceeded initial expectations. An additional bonus is that this Genetic Planner is able to solve the same problems as SINERGY [5] and use fewer generations to do so. These results are give further positive indication of the power of the Genetic Planner.

Logistics Domain

The Logistics domain is also a newer planning domain which takes its inspiration from package delivery companies. There are a number of different locations, forms of transport, and packages to be delivered. For instance within a particular city there may be a town centre, post office and airport. Each destination within the city can be driven to by a truck located in that city. However trucks can not drive between cities. Packages can be loaded and unloaded from a truck. Each city is connected by its airport between which aeroplanes can fly. Packages can also be loaded and unloaded from aeroplanes. The goal in this domain is to come up with a plan, which will move packages around from an initial location to a goal location via the trucks and aeroplanes.

The first problem used was the most basic one mentioned by the PRODIGY [11] system. This involves three cities, each with three locations and a truck, and two aeroplanes. The three packages start in PGH post office and two of them have to make there way to BOS post-office and the other to LA post office. The operators used are the same as ones in the BlackBox logistics-strips domain description file. This problem is unsolvable by the Genetic Planner. All the preceding problems were made more simple than this problem by involving less locations and packages. This particular problem is marked "LOG-7" in the following table.

An experiment was still considered to discover the relative effectiveness of the Genetic Planner on this problem domain. The standard experimental set-up was used. Each problem in the Logistics domain was run 10 times with the best and average number of solutions to the first correct solution recorded. In LOG-4 the Genetic Planner had to get the packages from the post office to the airport. LOG-5 required that the packages add to get inside particular aeroplanes. For LOG-6 the Genetic Planner had to get the packages to the correct city airports. Finally in LOG-7 the Genetic Planner had to get the packages to the right post offices.

Problem	Description	Average	Best	Action Space
LOG-1	1 city, 3 locs, 3 packs, 1 truck	0.9	0	24
LOG-2	3 cities, 3 locs, 3 packs, 2 planes	3.8	2	30
LOG-3	2 cities, 6 locs, 1 pack, 2 trucks, 2 planes	6.75 (6 failures)	6	36
LOG-4	3 cities, 9 locs, 3 pack, 3 trucks, 2 planes	8	4	120
LOG-5	3 cities, 9 locs, 3 pack, 3 trucks, 2 planes	15 (9 failures)	9	120
LOG-6	3 cities, 9 locs, 3 pack, 3 trucks, 2 planes	X	X	120
LOG-7	3 cities, 9 locs, 3 pack, 3 trucks, 2 planes	X	X	120

These results are interesting as both Muslea and Spector produced encouraging results for the Genetic Planner. The Genetic Planner is unable to solve the simplest problem provided by the PRODIGY system and even worse it is only able to solve simpler instantiations of this problem. The problem arises from the fact that the Genetic Planner must string together several actions just to reach one goal. As there are only three goals in LOG-7, this means each goal is worth a large amount of fitness. For example, to move a package from one city post office to another takes nine actions. It is extremely unlikely that these nine actions will be present in the initial population in the correct order. We can view such a candidate as building block of a successful plan. This building block will never be constructed by the algorithm. We attribute this problem to the fitness function used. Only counting the number of goals achieved will tell us nothing about how far the package is from the goal location. Therefore the fitness function is not giving enough feedback to the algorithm. The probablility that these nine action are present in the initial population with a particular candidate is also highly unlikely. A quick improvement to the fitness function is to include a small proportion of the overall fitness value to those candidates that contain actions which fire. The new fitness function incorporating a working component, overcomes the deficiency of the goal counting fitness function slightly as the Genetic Planner puts together large chunks of working actions in the blind hope that they will achieve a goal.

A possible solution to the problem would be to devise a new domain dependent fitness function that computes the number of steps the package still has to go in reaching its goal. This would allow the fitness function to give more useful feedback to the Genetic Planner as those plans, which move the packages towards the goal would receive more fitness. This approach is taken by Gair [2] with good results. His Genetic Planner, which also used automatically defined functions, was able to solve prob20 and prob28 in the BlackBox distribution. Another possible alternative solution would be to recognise landmarks as use this information to feed into the fitness function [8].

One of the original project goals was to try the Genetic Planner on a variety of Problem Domains. This has allowed the presentation of new types of problems, which require new types of progress to proceed. The Logistics Domain does this admirably. The Genetic Planner performed poorly on this problem domain and is the first negative result for genetic planning. The domain caused a rethink on the effectiveness of the current best fitness function. This domain presented a new type of planning domain where goals are placed many actions away from the initial state. What is therefore required is a new domain independent fitness function. This fitness function may defined automatically in some way by the Genetic Planner or may be of human creation.

Future Work

During the implementation of the Genetic Planner many avenues of further work were brought forward. We believe the most exciting of these, and first suggested by [9], would be to make the Genetic Planner search for general plans for particular domains. These general plans could then be shrunk done to become general planning rules about a domain. Using the general rules would allow the planner to make shortcuts, much in the same way humans do, and therefore reduce the amount search required to solve a particular problem. These shortcuts could then be incorporated into a knowledge base and then perhaps be extended and generalised so they can be applied to a number of domains. One of the shortcomings of most planning systems is that they learn nothing about a problem or a domain. Each time they solve a problem they solve it from scratch. We believe it would be interesting to see what general rules could be learned from different problem domains and to store this knowledge somewhere to better enable the Genetic Planner or even other planners.

Other areas of future work are extending the Genetic Planner to work on other areas of planning such as: dynamic environments, adversarial planning, hierarchical planning and so on. This is necessary as the classical domain is too constrained to be of any practical use. Therefore it must be known whether Genetic Planners scale up to the more expressive problem domains.

Finally there are many short-term improvements that can be made to the Genetic Planner that is presented here. Firstly the initial candidate generation method can be vastly improved to produce partially correct plans rather than random ones. This would be an attempt at giving all the building blocks the Genetic Planner requires in order to construct a solution, instead of making these building blocks from scratch. A possible solution to this problem could come from Graphplan as it has a computationally inexpensive way of expanding the planning search space. The initial population could be filled by random walks through the graph produced by Graphplan.

The next area to apply research would be to the problem domain representations and the fitness functions used. The representation of problems should be improved to be more faithful to the problem domain it represents. In addition to this, domain dependent fitness functions could then be created to exploit these representations and then give better feedback to the Genetic Planner. An improved domain independent fitness function also must also be found to improve the performance of the Genetic Planner on all problems and all problem domains. Different fitness functions could be implemented that try to find different attributes of plans. For instance a fitness function may try to find the cheapest plan in terms of using the least expensive actions, or a fitness function may be devised to produce the most robust plan. Finally another area that needs further work is the crossover function. Crossover can be better implemented, perhaps by taking inspiration from biological homologous crossover.

Conclusions

Genetic Planning offers much as a potential solution to the planning problem. Though not always finding the optimal solution for the most difficult problems, it has the potential to offer solutions to difficult problems that are out of reach of current planning technology. This is demonstrated by the results of the Blocks World domain.

In addition to this the Genetic Planner confirms some of the results gained from Muslea's SINERGY [5]. The Genetic Planner was not only able to solve the same problems as SINERGY but used less generations to do so. There is obviously much in the specific implementation of the Genetic Planner.

Finally the Logistics Domain showed that additional thought is required in order for the Genetic Planner is to solve more challenging problems. Bear in mind that the most simple domain independent fitness function and representation was used in attempt to solve the problem. It is hoped that improvements to the Genetic Planner as a whole will allow solution to this particular problem domain.

Genetic Planning has much to offer, as it allows a new and feasible domain independent way of tackling the planning problem. It also offers insight to the Genetic Programming discipline where the programs being sought for have to be linear in nature and have high dependencies between programming constructs.

Acknoledgements

The first author would like to thank the EPSRC for support via a quota studentship.

References

[1] Banzhaf, W. Nordin, P. Keller, R.E. and Francone, F.D. *Genetic Programming An Introduction* San Francisco CA Morgan Kaufmann Publishers, 1998

[2] Gair, J. *Genetic Programming with ADFs in the logistics planning domain* MSc AI Dissertation, Division of Informatics, University of Edinburgh, 2000

[3] Kautz, H. and Selman, B. *Unifying SAT-based and Graph-based Planning* Proc. IJCAI-99, Stockholm, 1999

[4] Koza, J.R. *Genetic Programming* Cambridge MA, The MIT Press, 1992

[5] Muslea, I. *A General Purpose AI Planning System Based on the Genetic Programming Paradigm* Late Breaking Papers at Genetic Programming 1997 Conference

[6] Muslea, I. *SINERGY: A Linear Planner Based on Genetic Programming* Proceedings of the 4th European Conference on Planning, 1997

[7] Muslea, I. *A General Purpose AI Planning System Based on the Genetic Programming Paradigm* Proceedings of the World Automation Congress, 1998

[8] Porteous, J. and Sebastia, L. *Extracting and Ordering Landmarks for Planning* Poster Presentation, PLANET Planning Summer School, 2000

[9] Spector, L. *Genetic Programming and AI Planning System* AAAI-94

[10] Weld, S. *An Introduction to least commitment planning AI* Magazine 15(4), 1994.

[11] Veloso, M. Carbonell, J. Pérez,A, M. Borrajo, D. Fink, E. Blythe, J. *Integrating planning and learning: The PRODIGY architecture* Journal of Experimental and Theoretical Artificial Intelligence, 7(1):81-120, 1995.

[12] Westerberg, C.H. *An investigation into the use of using Genetic Programming to solve Classical Planning Problems* 4th Year CS/AI Project, Division of Informatics, University of Edinburgh, 2000

[13] Blackbox: http://www.research.att.com/~kautz/blackbox/index.html