Evaluation of Genetic Algorithm on Grasp Planning Optimization for 3D Object: A Comparison with Simulated Annealing Algorithm

Zichen Zhang
Department of Electrical and Computer Engineering
Dalhousie University
Halifax, NS B3J 2X4
Email: zichen.zhang@dal.ca

Jason Gu
Department of Electrical and Computer Engineering
Dalhousie University
Halifax, NS B3J 2X4
Email: jason.gu@dal.ca

Jun Luo
School of Mechatronic Engineering and Automation
Shanghai University
Shanghai, China
Email: luojun@shu.edu.cn

***Abstract*—Grasp planning based on geometrical information of objects can be approached as an optimization problem where a hand configuration that indicates a stable grasp needs to be located in a large search space. In this paper, we study the applicability of genetic algorithm (GA) on grasp planning optimization of 3D objects. The details are given on the selection of operators and parameters. Different sampling methods in the implementation of crossover and mutation operators are tested. A quantitative analysis including the comparison with random planner and simulated annealing (SA) method is performed to evaluate the performance of the GA based planner. *GraspIt!* simulator [1] is used for implementing the proposed algorithm and as the test environment. Two different quality metrics are considered. The result shows that GA is a robust method in the field of grasp planning. And the GA planner outperforms the SA planner in both pre-grasp quality and stability of the final grasp.**

I. Introduction

Grasp planning based on the geometrical information can be approached as an optimization problem: to search for a stable grasp either from the contact space of the object or from the space of possible hand configurations. The first approach is classified as *synthesis approach*, while the second one classified as *heuristic approach* [2].

The goal of *synthesis approach* is to find the contact points on the surface of the object which indicate stable grasps. This type of approach generally requires a precise placement of fingers on an object, which may not be achievable in practical applications. And also it is not the natural way of grasping an object from a human's perspective.

In our work, we focus on *heuristic grasp planning*, a grasp action starting with "pre-grasp"-a hand posture close to but not in touch with the object-inspired by the fact that humans unconsciously preshape the hand before the actual grasp [3]. It defines the starting posture and approach direction of the hand. After the *preshape* is determined, the hand is moved along the direction toward the object until in contacts. Then fingers are closed to conform to the surface of the object to complete the whole grasp action. The pre-grasps can be obtained in this way: for simple gripper, by some simple heuristic rules based on the object shape or for more dexterous hand, by searching for hand poses that will be likely to yield grasps with good quality.

Both approaches can be considered as optimization problems. But the solution space is too vast to search in an effective manner. And it is discontinuous since there are some surface points or hand configurations that we want to avoid. Also the quality measures for evaluating the stability of a grasp are often non-linear. As a general-purpose optimization method, Genetic Algorithm (GA) is widely used to tackle with this type of problem.

In the literature, some work has been done in applying genetic algorithm to synthesis grasp planning. A. Chella *et al.* proposed a hybrid method of GA and neural network to planar object grasping [4], where a dataset is first generated off-line by using genetic algorithm and then trained by neural networks for the purpose of real-time application. This method only dealt with 2D objects in superellipses shape. N. Daoud [5] used GA to find grasps for 3D objects with an LMS mechanical hand. It had not been tested on other hand types and only limited information was given on the performance of the algorithm.

For the *heuristic grasp planning*, although GA has been mentioned as a possible method for optimization in this scenario [1], not much work has been done in applying genetic algorithm to this problem. The first relevant work dates back to 1998, when J.J Fernandez *et al.* proposed a genetic approach to find good grasps with three-fingered robot hands [6]. It was restricted to a certain hand-object placement and several pre-defined 3D objects with regular shapes and three-fingered hands. In [7], GA is applied to search for hand position and orientations. But the preshape of the hand is fixed for certain objects, not considered in the optimization process. And the performance of the GA planner is only compared with a random planner. In addition, no quantitative result has been addressed in the literature regarding the performance of the GA based grasp planner as opposed to other optimization techniques and few details have been found on the parameter selection. These make it difficult to compare with planners based on other algorithms.

In this paper, we study the effect of applying GA on *heuristic grasp planning* in order to fill the gap of current research and gain a better understanding of GA's applicability in the context of grasp planning. We carefully choose the operators and parameters of GA taking into account the characteristics of the solution landscape in this grasp planning problem. The effects of different sampling methods in the implementation of GA will be investigated as well. And the proposed GA grasp planner is applied on different sets of hand-object including 3D objects in different shapes and hand models with different number of DOFs to examine its performance and robustness. Comparison with other algorithms such as a simple random algorithm and simulated annealing algorithm (SA) are conducted for further evaluation of the GA planner. The execution time of SA and GA planners are presented for a fair comparison of their performance. And two different quality metrics will be considered.

This paper is organized as follows. Section II formulates the optimization problem in grasp planning and describes the quality metrics used in this work. The components and important concepts in designing a GA based grasp planner are discussed in Section III. In Section IV, the performance of the GA planner is examined with quantitative analysis. Section V concludes the paper and outlines the future work.

II. Problem Formulation

The first step of the grasp planning is to find a good pre-grasp that is expected to yield a force-closure final grasp. Then the final grasp will be executed and stability will be checked as the second step. The quality metric proposed in [8] is used to evaluate the quality of a *pre-grasp*:

$$Q_{pre} = \sum_{all\ contacts} \delta_i \tag{1}$$

where δ_i is a measure of the distance between the desired contact locations on the hand and the object. The contact locations on the hand are selected to be on the fingers and palms in our study to create a power grasp, which is more suitable on human-made objects than pinch grasp using fingertips only [9]. This quality measure assumes that the closer the hand is from the object, the better potential it has to give a stable power grasp on the object. For details, the reader is referred to [8].

The lower the Q_{pre}, the closer the hand is from the object, and the better the pre-grasp is considered to be in yielding a force-closure final grasp(although not always the case, as shown in the results section). The optimization goal is to find a pre-grasp that can minimize this pregrasp quality measure. To reduce the dimensionality and make the optimization speed faster, we consider the grasp planning in *eigengrasp* space, where the high-dimensional space is projected to a low-dimensional control space while maintaining sufficient information needed for finding stable grasps [8]. Therefore, the grasp planning problem in the configuration space of the hand can be represented as:

$$\operatorname*{argmin}_{p,w} Q_{pre}(p,w), \text{ subject to} : p \in \Re^b, w \in \Re^6 \tag{2}$$

where $Q_{pre}(p,w) : \Re^d \mapsto \Re$ is the objective function to be minimized over the variable space of dimension $d = b+6$. b is dimension of the eigengrasp space, p is a vector representing the hand posture, and w is a vector of the position and orientation of the wrist.

TABLE I: Variable List

Property	Name	Definition	Range
Position	Tx	x-coordinate	[-250,250]
	Ty	y-coordinate	[-250,250]
	Tz	z-coordinate	[-250,350]
Orientation	θ	angle between the z-axis and the axis	[0,π]
	ϕ	angle between the projection of the axis on x-y plane and x-axis	[-π,π]
	α	rotation angle around the axis	[0,π]
Eigengrasp	EG[0,..,b]	amplitude along the eigengrasp dimension	[-4,4]

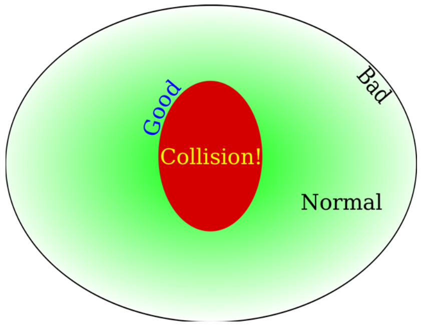


Fig. 1: The solution landscape

The variables are listed in Table I. A larger range of Tz is chosen for tall objects. Axis-Angle representation is used for the 3D rotation. Note that the pregrasp is only the starting stage of a complete grasping action. To assess the stability of the final grasp, we use ε quality and v quality proposed in [10]. For force-closure grasps, $0 < \varepsilon < 1$, $v > 0$. The larger these two quality measures are, the more stable a grasp is.

III. Genetic Algorithm on Grasp Planning

In this section, we will apply GA on grasp planning. We start with a close look at the search space. The selection of operators and parameters are then investigated in detail. To understand and overcome the sampling bias caused by crossover and mutation operators, we also test different types of sampling methods.

A. Solution Landscape

The GA planner needs to be designed according to the characteristics of the search space, which is the space of all possible solutions to a problem.

First we need to decide on the encoding for a solution. As the variables in this problem take real value, we naturally adopt floating-point representation. Each chromosome can thus be represented as:

$$chromosome = < Tx, Ty, Tz, \theta, \phi, \alpha, EG_0, EG_1, \ldots > \tag{3}$$

The position and orientation of the hand is defined in terms of the contact space of object. We define the range of the variables so that possible solution space is around the object, as shown in Table I. The solution landscape is illustrated in Fig. 1. The red area shows the space taken up by the target object. The *pre-grasp* is considered to be illegal if the hand and the object get in touch. The legal solutions fall in the space outside the object.

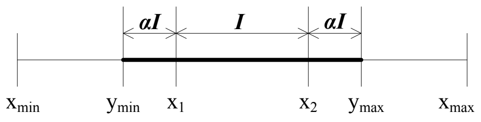


Fig. 2: BLX-α

The closer to the object, the better the pre-grasp is. "Good" denotes the space that gives the best pre-grasps. As it extends outward, the quality goes down, denoted by "Normal". In the solution space close to the boundary (e.g. when the position variables take values to the minimum or maximum, the hand is far away from the object), the pre-grasps has a very low quality, denoted as "Bad". Although "Bad" pre-grasps are not considered to be illegal, we want to avoid them. "Normal" and "Bad" do not indicate particular quality value. They are just two terms used to informally show the transition trend of the solutions. In the grasp planning, we will only search for solutions that do not collide with the object. That is, if any illegal solution is obtained during any step of the GA, it will be dropped and that GA step will be repeated until a legal solution is found. This makes the solution space discontinuous. Imagine the action of moving the hand from inside the object to outside, there will be a sudden transition from the worst grasp to the best. It addresses one of the difficulties for an optimization algorithm to find the best solution.

Good solutions are not distributed evenly in the search space. For the three position variables denoting the position of the the wrist, we want the search to be focused on the center area, where the hand is closer to the object. There is no preferred range for the orientation and eigengrasp variables. The possible solutions should be searched evenly throughout the entire range.

B. Operator Selection

We use Tournament Selection with a tournament size of two as the *parent selection* technique. For crossover operator, we employed BLX-α operator proposed in [11], which is defined as: suppose we have two parents x_1, x_2, children y_1, y_2 are uniformly chosen from the interval $[C_{min} - I\alpha, C_{max} + I\alpha]$ at random, where $C_{min} = min\{x_1, x_2\}, C_{max} = max\{x_1, x_2\}, I = C_{max} - C_{min}, \alpha \in [0, 1]$. This can be illustrated in Fig. 2. α is normally set to a number bigger than 0, to make the children generated span a slightly larger space than the parents, which from a statistical perspective compensates for the shrinking of the solution space over the generations [12]. The effect that the solution space will be attracted towards a certain area preferred by the search operator is also called the search bias [13]. There will be further discussion later.

Gaussian Mutation is applied as the mutation operator. A new gene value is obtained by adding to the current value a number drawn from a Gaussian distribution $N(0, \sigma)$, where σ is a user-specified parameter [14]. Let

$$\sigma_i = K_\sigma \cdot r_i \tag{4}$$

, where r_i is the range of the value of $gene_i$. We want to select the parameter K_σ to make sure that the possible solution can cover the full range. Since the value range for a Gaussian distribution is approximately equal to 6σ, we choose $K_\sigma = 0.2$ as an initial value. Larger values of K_σ will be tested in order to find the most appropriate value.

In addition, we adopt the *generational model* for survivor selection, i.e., the entire population are replaced by their offspring at each generation. To summarize, the operators are listed in Table II.

TABLE II: Operator List

Operators	Method
Representation	Floating-point Numbers
Parent Selection	TournamentSelectionWithoutReplacement, $T_s = 2$
Crossover	BLX-$alpha$
Mutation	Gaussian Mutation, $K_\sigma = 0.2$
Elitism	Two Elitists, added to the next population
Survivor Selection	Generational Model

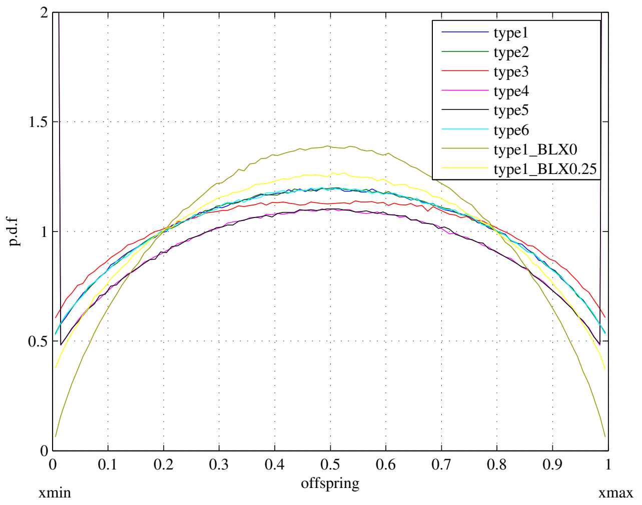


Fig. 3: BLX-α with six sampling methods

C. Sampling Methods

As discussed earlier, there is an inherent bias caused by the crossover operator, that the search will go toward a certain area rather than the whole space. It is necessary to investigate the bias of BLX-α crossover operator to find out if this bias is beneficial for the search.

In BLX-α crossover, children solutions are generated by sampling from an extended area around the parents. In [15], the search bias was examined with three sampling methods. We will extend their work with six sampling methods. Details of each method are described in the Appendix.

We consider one dimensional search without loss of generality. This search space is given by

$$X = \{x \quad ; \quad x_{min} \leq x \leq x_{max}\} \tag{5}$$

We run the BLX-0.5 operator on the randomly generated parents x_1, x_2, using the above six sampling methods. Each one is run for 5,000,000 times. The probability density function (p.d.f.) of offspring y generated with each method are shown in Fig. 3. BLX-0 and BLX-0.25 with type1 sampling are also included.

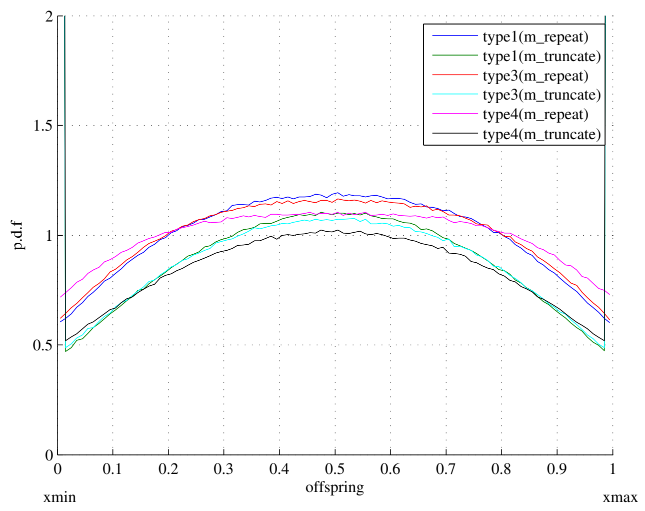


Fig. 4: Two different mutation sampling methods with different sampling for BLX-0.5, "m_repeat" means repetition sampling for mutation, "m_truncate" means truncation sampling for mutation

From the figure, we can see that the BLX-α operator has an inherent bias towards the center of the search space. Comparing these six sampling methods, we can see that the p.d.f of type(1,2,6) are almost identical, and type(4, 5) are similar as well. We will treat them as equivalent. Type3 sampling produces solutions more evenly distributed in the search space. Type(1,2,6) has highest p.d.f. on the center while the boundary gets a lowest p.d.f. The solutions of type(4,5) have a very high p.d.f near the boundary because of the truncation. This leads to the lowest p.d.f in the remaining search space. We would prefer type(4,5) only when the global optimum is very close to the boundary of the search space. Type3 sampling would be the one preferred if no *a priori* information is known about the solution space because of its less bias.

Note that the above probability density functions are generated empirically. For the theoretical proof on the p.d.f of offspring produced by BLX operator, readers are referred to Appendix A in [13].

In the discussion above , we focused on crossover operator only. In fact, in the implementation of Gaussian Mutation, there also can be two sampling methods: one is repeating until the children are in the feasible space, the other is truncating the children to the limit of the feasible range. We call these two methods the "repetition method" and "truncation method" respectively. The effect of them on mutations are illustrated in Fig. 4. The solutions of the truncation method concentrate on the boundary while other areas are less likely to be searched as opposed to the repetition method.

IV. Results and Discussion

A. Test Platform

The algorithm is implemented in C++ under a modified version of *GraspIt!* [1] which runs in ROS framework [16]. All the tests are performed in this modified *GraspIt!* simulator, on a desktop computer with a AMD Athlon II Quad-core 2.9Ghz CPU and 6G RAM. The objects used for the tests are imported from the *Household Objects and Grasps Data Set* [17]. The Barrett Hand and the Human Hand model released with the *GraspIt!* simulator are employed in this study. The DOFs and the eigengrasp (EG) space of the hand models are listed in Table. III.

TABLE III: Dimension of the search space

Hand Model	DOFs and Eigengrasp Space	GA Encoding Length
Barrett	4DOFs $\mapsto$ 2 EG	8 genes
Human Hand	20DOFs $\mapsto$ 6 EG	12 genes

B. Parameter Tuning

Three parameters are typically considered in the parameter tuning of GA:

- n : Population size
- p_c : Probability of crossover
- p_m : Probability of mutation

In our case, we also decide on which sampling method to use. And we will tune α in the BLX operator and K_σ in the Gaussian Mutation operator.

There is no general theory on parameter selection. The optimal parameters are generally problem-dependent. In practice, parameters are often empirically tuned until satisfactory results are obtained. To tune the parameters, the program is executed with a Barrett Hand grasping a glass. n, p_c and p_m will be tuned in the range: $n = 50-100$, $p_c = 0.5-1.0$, $p_m = 0.01-0.5$. Some preliminary tests show that the GA planner usually does not produce better solutions after 5,000 generations with a population size of 50 or 100. Optimization with each set of parameters is performed over 5,000 generations and the best pre-grasp found is saved as well as the running time. To account for the stochastic nature of GA, each test is repeated five times. The best pre-grasp qualities are taken as the average from the five runs, denoted as "Average" in the tables.

The best parameter found for this planner is $n = 100$, BLX-0.5 with type1 sampling, Gaussian Mutation with $K_\sigma = 0.2$ and repetition method. Results are presented in Table. IV. Note that in this table, we also show the best pregrasp quality found from the five runs, which is put in the parentheses next to "Average". "STD" is the estimated standard deviation. The best average solution achieved is marked in red, while the best solution is marked in blue.

We will use $p_c = 0.8, p_m = 0.1$ throughout the following tests. This set of p_c, p_m is chosen since it finds the best solution and also determines a good average best solution. It should be recognized that, the search space is different when either the hand or the object is changed. The parameters we chose may not be applicable for other hand-object combinations. Thus the robustness of the GA planner is very important in grasp planning. Further tests will be presented in the following sections.

TABLE IV: BLX-0.5 with type1 sampling, $K_\sigma = 0.2$ with repetition method for mutation

Repetition Sampling		p_m											
		0.01		0.1		0.2		0.3		0.4		0.5	
		Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
p_c	0.5	26.815(21.408)	5.349	17.982(14.626)	6.821	16.098(15.145)	0.859	18.69(16.568)	1.673	18.815(16.443)	2.304	18.978(16.095)	2.083
	0.6	31.07(22.53)	5.196	24.997(17.697)	6.166	16.846(15.048)	2.133	18.141(15.039)	1.796	18.976(15.058)	2.62	20.344(19.419)	0.979
	0.7	26.915(15.918)	6.81	25.668(14.885)	6.079	17.726(16.264)	1.363	21.124(17.531)	4.891	18.792(16.916)	2.257	20.068(17.475)	1.723
	0.8	26.752(19.639)	5.692	**16.96(13.678)**	**6.629**	18.327(16.909)	1.687	20.306(17.657)	2.414	21.594(19.206)	2.083	20.853(17.717)	3.204
	0.9	20.189(15.906)	5.213	23.235(14.039)	8.149	15.828(15.428)	0.259	19.953(16.827)	2.385	19.995(18.763)	0.973	22.74(20.27)	2.112
	1	21.723(14.738)	6.912	**15.64(13.988)**	**2.051**	19.037(16.306)	3.207	19.415(17.549)	1.481	21.136(18.151)	2.608	23.739(21.234)	1.622

C. Performance

We run the GA planner with the best parameters we selected from the previous section. The on-line performance and off-line performance proposed by De Jong [18] are used to monitor the evolution of the quality of the grasps over generations and evaluate the convergence performance of GA. On-line performance at generation t is an average of the best from each generation in the past. Off-line performance keeps track of the best solution $Q(best)$ up to each generation and is calculated by taking an average of $Q(best)$ of the past generation at generation t. The on-line and off-line performance over

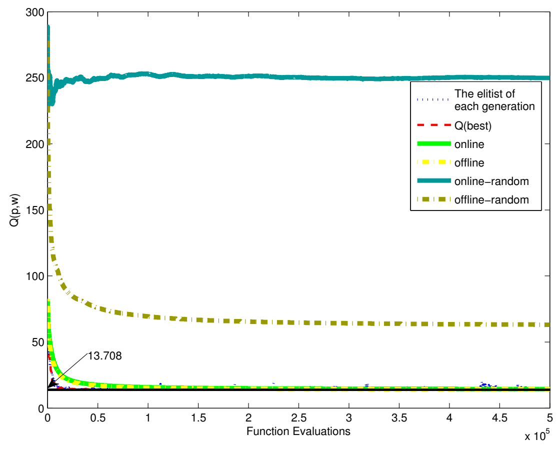


Fig. 5: The on-line and off-line performance of the genetic algorithm and random planner

5,000 generations as well as the elitists and the $Q(best)$ of each generation are shown in Fig. 5. In this plot, we also include the on-line and off-line performance of a random planner. This planner simply generate a random number in the search space for each variable at each step. For a fair comparison, the random planner is executed for $5,000 * 100 = 500,000$ function evaluations. And the results for both planners are presented in the figure over function evaluations rather than generations.

The figure shows that the population of the planner evolves rapidly over the first 200-250 generations and grows slowly afterwards. In the later stage, the population reaches a stable status. Most individuals are within a small space around the elitist. During this time period, crossover hardly produces new individuals and is not efficient in the search of the solution space. However, the diversity is maintained by mutation so that the rest of the configuration space still gets the chance to be searched over, which is indicated in the blue line. Compared with the simple random planner, we can see that GA is far more efficient. That is the reason we do not want to use a high mutation probability for GA, since that will make it act like a random algorithm.

D. Comparison with Simulated Annealing Planner

Quantitative results on the performance of simulated annealing on grasp planning were reported in [8] and implemented in the original release of *GraspIt!* simulator. To further assess the performance of the proposed GA grasp planner, we test both GA and SA planner on the same sets of hand-object combinations and compare their performance. The SA planner from the original *GraspIt!* is used in our study as the benchmark since it is very well tuned. In our test, it is performed over 70,000 iterations as suggested in [8] and the GA planner is terminated at 5,000 generations, which is 500,000 function evaluations. Each test is repeated five times and the best pre-grasp quality results are averaged. These results are given in Table V. And the best pre-grasps found for both planners on each hand-object combination are shown in Fig. 6.

The results shows that GA planner outperforms SA planner in most cases in terms of the best solution. And it is robust to different hand-object combinations. The average execution time of the two algorithms is listed in Table VI. The SA planner performs faster than GA planner. This shows that GA is better in finding global optimal pre-grasp with a sacrifice in calculation speed. In fact, the optimization in grasp planning usually finds its application in off-line use, such as building a database [19]. Thus the difference in execution time is not a big concern in this work. However, for a fair comparison, we want to see what happens if the two algorithms are given the same amount of time.

We run the GA planner with the same time as the average time of SA planner listed in Table VI. The result is also obtained from five runs on each hand-object combination, shown together with result of SA planner from Table V in Table VII. The better average pre-grasp quality obtained for each hand-object from the two planners is marked in red. GA planner performs better in five out of eight cases. This shows that even given the same amount of time, the performance of GA is comparable to that of SA. We recognize that it is helpful to have GA as another option in the grasp planning task. In applications such as building a database, we can run both GA and SA planners and save the better result into the database.

The final grasps resulting from the pre-grasps in Fig. 6 are executed and shown in Fig. 7. “e” and “v” refers to the ε quality and v quality. The object is set to transparent to show the contact between the hand and object. Note that, the quality of pre-grasp is an informal estimate of the final grasp quality. But they are not equivalent. A good pre-grasp may result in a

TABLE V: Statistics of the best pre-grasps found from both planners

Planner Type	Glass				Bottle				Mug				Spray Bottle			
	Barrett		Human		Barrett		Human		Barrett		Human		Barrett		Human	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
SA	13.968(13.717)	0.297	12.032(11.382)	0.574	15.317(13.22)	2.155	10.005(8.75)	1.422	14.847(11.104)	4.115	11.999(10.885)	1.561	14.91(14.364)	0.494	9.221(7.138)	2.633
GA	13.819(13.421)	0.689	11.84(10.774)	1.15	11.773(11.56)	0.189	9.017(8.732)	0.312	13.055(11.194)	1.694	10.387(9.479)	0.807	12.945(10.34)	2.706	9.722(8.972)	1.11

TABLE VI: Execution time of the GA and SA planners

Planner Type	Execution Time(seconds)	
	Barrett	Human Hand
SA (70,000 iterations)	125	183
GA (500,000 function evaluations)	226	272

TABLE VII: Statistics of the best pre-grasps found from both planners given the same running time

Planner Type	Glass				Bottle				Mug				Spray Bottle			
	Barrett		Human		Barrett		Human		Barrett		Human		Barrett		Human	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
SA	13.968(13.717)	0.297	12.032(11.382)	0.574	15.317(13.22)	2.155	10.005(8.75)	1.422	14.847(11.104)	4.115	11.999(10.885)	1.561	14.91(14.364)	0.494	9.221(7.138)	2.633
GA(Same Time with SA)	13.89(13.567)	0.317	11.983(10.866)	0.998	13.131(11.951)	1.709	10.681(8.814)	1.172	15.707(14.807)	0.814	10.566(10.174)	0.374	13.793(11.976)	1.597	10.801(9.622)	0.686

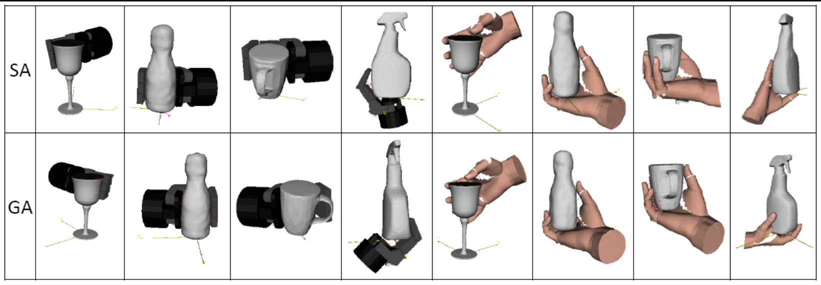
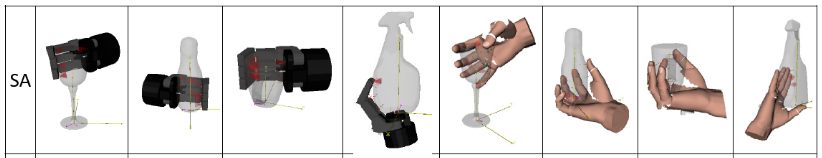
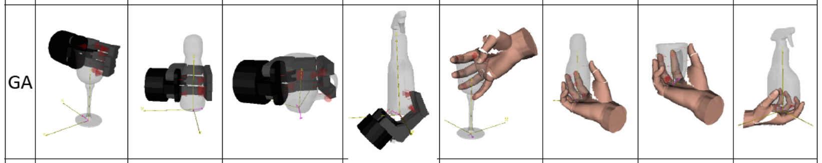


Fig. 6: Best pre-grasps found by GA and SA planners



SA	e:0.008 v:0.002	e:0.0129 v:0.0074	e: 0.0275 v:0.00679	e:-1 v: 0.000384	e:0.00479 v:0.374	e:0.0913 v:0.247	e:0.244 v:0.682	e:-1 v:0.0824



GA	e: 0.0122 v:0.0157	e: 0.0329 v:0.00341	e:-1 v:0.0414	e:-1 v: 0.000285	e:0.06115 v:0.516	e: 0.0916 v: 0.267	e:0.263 v:1.278	e:0.0896 v:0.113

Fig. 7: The corresponding final grasps and the quality

non F-C grasp and a better pre-grasp may leads to worse final grasp as indicated in Fig. 7.

E. Grasp Planning of Final Grasps

The pre-grasp quality metric that we used is fast to compute but it has a limitation. It was focused on forming an enveloping grasp around the object, which from a stability standpoint may not be a force-closure grasp. And a pre-grasp with a low (better) quality is not necessarily an enveloping grasp. For instance, if the hand has all the fingers fully opened, and is positioned very close and parallel to the surface of the object (like the spray bottle used in our study), then the hand is so close to the object that the it may yield a very good pre-grasp quality, but the hand is not even enveloping the object.

As mentioned earlier, in off-line applications, the speed of the grasp planning method is not as important as the stability of the grasp that it can find. To better find force-closure final grasps and to further evaluate GA's applicability in grasp planning, we propose planning the pre-grasps using the quality of the final grasp directly. Instead of using pre-grasp quality as the objective function, each pre-grasp is evaluated with its corresponding final grasp so that the stability of the solution is guaranteed. For every pre-grasp found by the optimization algorithm, we move the hand along the approaching direction defined by the pre-grasp by maximum 50mm until the hand is in contact with the object and then close the fingers to complete the final grasp. If no contact is found in this 50mm distance, the hand is moved back to its intial position and the fingers are closed. Then the ε quality and v quality can be obtained. We use a combination of these two quality measures to evaluate the final grasp, which was originally found in the Graspit! release:

$$Q_{final} = -(100\varepsilon + 30v) \tag{6}$$

This Q_{final} is used as the objective function of the optimization. The negative value is taken so that it is a minimization problem, consistent with the pre-grasp quality metric we used in previous sections. We use ε as the primary quality measure. It gets more weight than v quality.

Since the execution of the final grasp takes a lot of computation power, both GA and SA algorithms run very slow on this problem. We run both planners with a time limit: 1,000 seconds for Barrett Hand and 1,500 seconds for Human Hand. The quality of the grasp found from five runs of both planners was summarized in Table VIII. And the best pre-grasps and their corresponding final grasps are shown in Fig. 8 and 9. Both planners are able to find *force-closure* grasps. GA clearly outperforms SA on all the objects when using Barrett Hand. For grasp planning with Human Hand, GA finds better solutions in three out of four objects tested. With this method, the stability of the grasp is guaranteed. And the result shows that GA is robust to different quality metrics in grasp planning.

V. Conclusions and Future Work

In this paper, we presented an extensive analysis on genetic algorithm in solving the optimization problem in grasp planning. The solution landscape were examined as well as the bias introduced by the crossover and mutation operator with different sampling methods. We implemented GA in *GraspIt!* simulator and tests were carried out to choose the appropriate parameters and sampling methods.

The quantitative results on a number of hand-object combinations indicate that genetic algorithm can be used to obtain a good pre-grasp and is robust to different solution space introduced by different hand or object. Compared to the grasp planner based on simulated annealing algorithm, the GA planner was superior in most cases in terms of the average best solution obtained. Even given the same amount of time, the performance of GA was comparable with that of SA. To overcome the limitation of the pre-grasp quality, we investigated the possibility of using another quality metric for this problem. In the optimization process, each pre-grasp was evaluated based on its corresponding final grasp with a stability quality measure consisting of ε quality and v quality. With this quality metric as the objective function, GA outperforms SA with the same execution time.

Future work will be focused on improving the robustness of the GA planner. We will apply adaptive methods to tune the parameters on-the-fly. And since GA is intrinsically parrellel, it would be interesting to investigate the performance of a parallel GA, utilizing the power of multiple-core computers. Furthermore, considering the resemblance between SA and GA, hybrid methods which combine them [20] to take the best from both worlds may largely improve the performance.

Appendix

For definition of Type1-3 sampling method, users are referred to [15].

(a) **Type4 sampling**

In type4 sampling, instead of truncating y_{min} and y_{max} as in Type2, we truncate y if it is out of the range $[x_{min}, x_{max}]$:

$$y' = y_{min} + u(y_{max} - y_{min}), \tag{7a}$$

$$y = \begin{cases} x_{min}, & \text{if } y' < x_{min}, \\ x_{max}, & \text{if } y' > x_{max}, \\ y', & \text{otherwise} \end{cases} \tag{7b}$$

(b) **Type5 sampling**

Type5 sampling is similar with Type3 except that it truncates the offspring in the last step rather than truncating y_{min} and y_{max}.

$$y' = \begin{cases} y_{min} + u_1(c - y_{min}), & \text{if } u_2 \geq 0.5, \\ c + u_1(y_{max} - c), & \text{if } u_2 < 0.5. \end{cases} \tag{8a}$$

$$y = \begin{cases} x_{min}, & \text{if } y' < x_{min}, \\ x_{max}, & \text{if } y' > x_{max}, \\ y', & \text{otherwise} \end{cases} \tag{8b}$$

where

$$c = \frac{(x_1 + x_2)}{2} \tag{9a}$$

$$u_1, u_2 : \text{uniform random number} \in [0.0, 1.0]. \tag{9b}$$

(c) **Type6 sampling**

Type6 sampling is a variation from Type3. There is no truncation done in this method. The steps are repeated until the offspring is in the feasible range.

$$y = \begin{cases} \begin{cases} y_{min} + u_1(c - y_{min}), & \text{if } u_2 \geq 0.5, \\ c + u_1(y_{max} - c), & \text{if } u_2 < 0.5. \end{cases}, & \text{if } x_{min} \leq y \leq x_{max} \\ \text{repeat sampling}, & otherwise \end{cases} \tag{10}$$

where

$$c = \frac{(x_1 + x_2)}{2} \tag{11a}$$

$$u_1, u_2 : \text{uniform random number} \in [0.0, 1.0]. \tag{11b}$$

References

[1] A. T. Miller, “Graspit!: A versatile simulator for robotic grasping,” Ph.D. dissertation, Columbia University, New York, USA, June 2001.

[2] D. Bowers and R. Lumia, “Manipulation of unmodeled objects using intelligent grasping schemes,” *Fuzzy Systems, IEEE Transactions on*, vol. 11, no. 3, pp. 320 – 330, June 2003.

[3] A. Miller, S. Knoop, H. Christensen, and P. Allen, “Automatic grasp planning using shape primitives,” in *Robotics and Automation, 2003. Proceedings. ICRA ’03. IEEE International Conference on*, vol. 2, Sept. 2003, pp. 1824 – 1829 vol.2.

[4] A. Chella, H. Dindo, F. Matraxia, and R. Pirrone, “Real-time visual grasp synthesis using genetic algorithms and neural networks,” in *Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence on AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, ser. AI*IA ’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 567–578.

TABLE VIII: Statistics of the best grasps found from both planners with Q_{final} as the quality measure. Better one indicated in Blue

Planner Type	Glass				Bottle				Mug				Spray Bottle			
	Barrett		Human		Barrett		Human		Barrett		Human		Barrett		Human	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
SA	-13.573(-14.56)	0.87	-64.103(-70.911)	9.278	-12.992(-14.123)	1.176	-58.32(-59.832)	2.489	-39.51(-43.992)	3.902	-198.512(-202.374)	6.304	-12.852(-15.905)	2.652	-35.337(-41.819)	7.244
GA	-15.576(-16.659)	1.073	-66.453(-71.21)	7.208	-13.918(-16.147)	1.969	-59.56(-61.986)	3.65	-42.266(-46.071)	4.749	-194.499(-214.712)	29.574	-14.462(-18.262)	3.291	-41.474(-49.492)	10.921

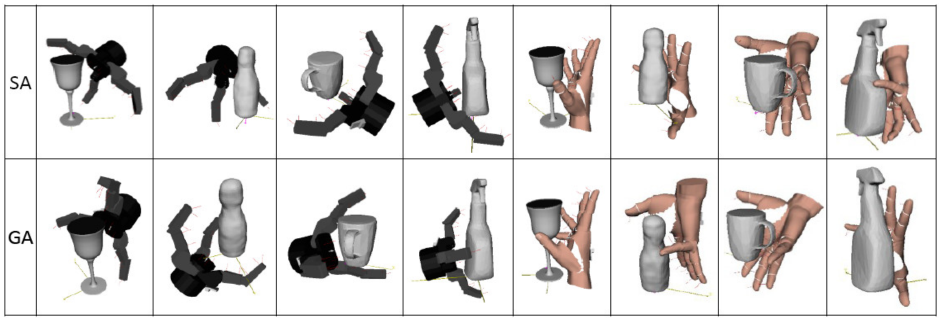
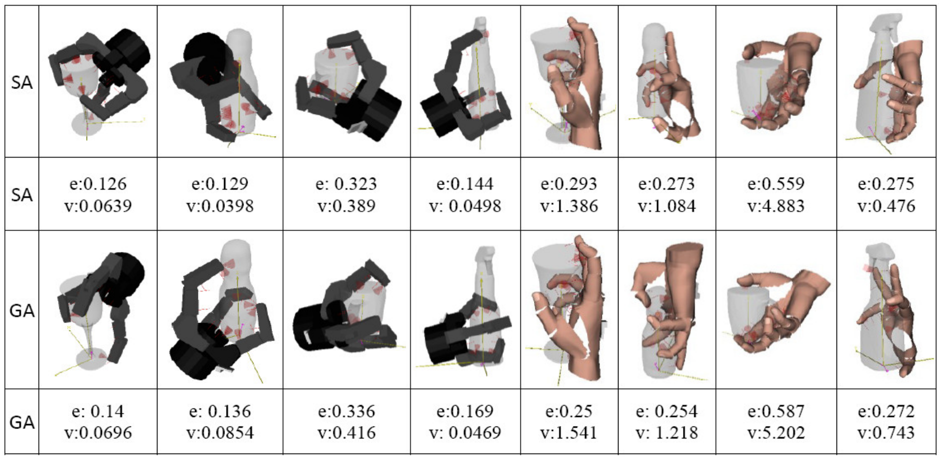


Fig. 8: Best pre-grasps found by GA and SA planners with Q_{final} as the quality measure



SA	e:0.126 v:0.0639	e:0.129 v:0.0398	e: 0.323 v:0.389	e:0.144 v: 0.0498	e:0.293 v:1.386	e:0.273 v:1.084	e:0.559 v:4.883	e:0.275 v:0.476
GA	e: 0.14 v:0.0696	e: 0.136 v:0.0854	e:0.336 v:0.416	e:0.169 v: 0.0469	e:0.25 v:1.541	e: 0.254 v: 1.218	e:0.587 v:5.202	e:0.272 v:0.743

Fig. 9: The corresponding final grasps obtained with Q_{final} as the quality measure

[5] N. Daoud, J. Gazeau, S. Zeghloul, and M. Arsicault, “A fast grasp synthesis method for online manipulation,” *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 421 – 427, 2011.

[6] J. Fernandez and I. Walker, “Biologically inspired robot grasping using genetic programming,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 4, May 1998, pp. 3032 –3039 vol.4.

[7] D. Berenson and S. S. Srinivasa, “Grasp synthesis in cluttered environments for dexterous hands,” in *IEEE-RAS International Conference on Humanoid Robots (Humanoids08)*, December 2008.

[8] M. Ciocarlie and P. Allen, “Hand posture subspaces for dexterous robotic grasping,” *The International Journal of Robotics Research*, vol. 28, pp. 851–867, 07/2009 2009.

[9] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, “Learning to grasp objects with multiple contact points.” in *ICRA*. IEEE, 2010, pp. 5062–5069.

[10] A. Miller and P. Allen, “Examples of 3d grasp quality computations,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1240 –1246 vol.2.

[11] Eshelman, “The CHC Adaptive Search Algorithm : How to Have Safe Search When Engaging in Nontraditional Genetic Recombination,” *Foundations of Genetic Algorithms*, pp. 265–283, 1991.

[12] H. Pohlheim, “GEATbx - genetic and evolutionary algorithm toolbox for use with matlab. http://www.geatbx.com/,.”

[13] Y. Yoon, Y.-H. Kim, A. Moraglio, and B.-R. Moon, “A theoretical and empirical study on unbiased boundary-extended crossover for real-valued representation,” *Inf. Sci.*, vol. 183, no. 1, pp. 48–65, Jan. 2012.

[14] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[15] S. Tsutsui and D. E. Goldberg, “Search space boundary extension method in real-coded genetic algorithms,” *Information Sciences*, pp. 133–3, 2001.

[16] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[17] M. Ciocarlie, C. Pantofaru, K. Hsiao, G. Bradski, P. Brook, and E. Dreyfuss, “A side of data with my robot: Three datasets for mobile manipulation in human environments,” *IEEE Robotics & Automation Magazine, Special Issue: Towards a WWW for Robots*, vol. 18, June 2011.

[18] K. A. De Jong, “An analysis of the behavior of a class of genetic adaptive systems.” Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1975.

[19] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, “The columbia grasp database,” in *IEEE Intl. Conf. on Robotics and Automation*, 2009.

[20] S. W. Mahfoud and D. E. Goldberg, “Parallel recombinative simulated annealing: A genetic algorithm,” *Parallel Computing*, vol. 21, no. 1, pp. 1–28, 1995.