THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2013-13

Separating the Searches of Bounded Rational Decision-Makers

Jeffrey W. Herrmann

Separating the Searches of Bounded Rational Decision-Makers

Jeffrey W. Herrmann
A. James Clark School of Engineering
University of Maryland
College Park, MD 20742
301-405-5433
jwh2@umd.edu

Abstract

In practice, when faced with a complex optimization problem, human decision-makers often separate it into subproblems and then solve each subproblem instead of tackling the complete problem. This paper describes a study that simulated small teams of bounded rational decision-makers ("agents") who try different approaches to solve optimization problems. In the "all-at-once" approaches, the agents collaborate to search the entire set of solutions in a sequential manner: each agent begins where the previous agent stopped. In other approaches, the agents separate the problem into subproblems, and each agent solves a different subproblem. Finally, in the hybrid approaches, the agents separate the problem but two agents will collaborate to solve one subproblem while another agent solves a different subproblem. In some cases, the subproblems are solved in parallel; in others, the subproblems are solved sequentially. The results show that the teams generally found better solutions when they separated the problem.

Introduction

In practice, human decision-makers often separate a complex optimization problem into subproblems and then solve each subproblem instead of tackling the complete problem. This approach is a natural strategy given the constraints that human decision-makers have. As part of an ongoing study of the effectiveness of this strategy, this paper presents the results of a study that considered some specific optimization problems, modeled teams of bounded rational decision-makers, and evaluated the quality of the solutions that these teams found using different

solution approaches. The study was intended to provide insights into the relationship between the quality of the solutions and the structure of the separations.

The remainder of the paper proceeds by reviewing the concept of bounded rationality, discussing how we modeled bounded rational decision-makers, and describing the separations studied. The paper then describes the instances used in the computational study, the design of the computational study, and the results of the study before concluding with some insights gained from this work.

Bounded Rationality

It is well-known that real-world decision-makers cannot optimize because of limits on their problem-solving capacity. This concept is known as "bounded rationality" (Simon, 1997a). Bounded rationality reflects the observation that, in most real-world cases, decision-makers have limited information and limited computational capabilities for finding and evaluating alternatives and choosing among them (Simon, 1997b; Gigerenzer *et al.*, 1999; March and Simon, 1993). A decision-maker cannot perfectly evaluate the consequences of the available choices. This prevents complete and perfect optimization.

Satisficing and fast and frugal heuristics are two models of bounded rationality (Gigerenzer *et al.*, 1999). Bounded rational decision-makers may search until they find something that meets their requirements (satisficing), or they may use fast and frugal heuristics that search a limited set of objects and information and make choices using rules that are easy to compute (and therefore quick). This study considered only this second type of bounded rationality.

When tackling a complex optimization problem, the decision-maker needs to determine values for various aspects of the solution. That is, he must make a decision. In practice, he does

not have complete information about the available alternatives and their impact on overall performance. Thus, he must employ some process for generating and evaluating alternatives.

March and Simon (1993) describe the general characteristics of human problem-solving in organizational decision-making. The first characteristic is that making a complex decision involves making a large number of small decisions. The second characteristic is that problem-solving has a hierarchical structure in which solving any problem goes through phases that, in turn, require solving more detailed subproblems. The general concept of separation is related to these two characteristics. The third characteristic is that problem-solving consists of searching for possible solutions (cf. Nutt, 2005). The fourth characteristic is that problem-solving includes screening processes that evaluate the solutions that are found. The fifth characteristic is that problem-solving has not only random components (such as finding and evaluating solutions) but also a procedural structure that allows it to yield good solutions. The proposed model of a bounded rational designer is motivated by these last three characteristics.

Wang and Chiew (2008) described human problem solving as a higher-layer cognitive process that can be considered as a search process, though it requires other cognitive processes such as abstraction, analysis, synthesis, and decision-making. Their model of a generic problem solving process is a search that iteratively generates and evaluates potential solutions.

Thus, we conclude that search is a valid model of bounded rational decision-making.

Herrmann (2010) took a similar approach but considered separations of a specific engineering design optimization problem as models of progressive design processes. His results indicated that well-designed progressive design processes are the best way to generate profitable product designs.

This paper considers a very different problem but addresses the same issue: how does the separation affect the quality of the solution? It also considers the best way to allocate the resources of a team of decision-makers to the subproblems in the separation.

Modeling Bounded Rationality

An important aspect of bounded rationality is that the resources and time available for problem-solving are limited. Consequently, the proposed model of a bounded rational decision-maker incorporates limits that will constrain the amount of time available for the search and the quality of a solution.

Gurnani and Lewis (2008) studied collaborative, decentralized design processes in which the models of the individual decision-makers (the designers) were based on the ideas of bounded rationality. In their model, the value chosen by each designer was determined by randomly sampling from a distribution around the (locally) "optimal" solution. This model was meant to represent the mistakes that designers make due to bounded rationality. Their results showed that incorporating bounded rationality led to more desirable solutions in a collaborative, decentralized design process in which the designers had different objectives and no way to coordinate their activities.

Herrmann (2010) presented a method for assessing the quality of a product design process by measuring the profitability of the product that the process generates. Because design decision making is a type of search, the method simulated the choices of a bounded rational designer for each subproblem using search algorithms. The searches, which were limited to a fixed number of iterations, had random components (either randomly selecting a solution or randomly moving to a point near the existing solution) and a procedural structure to keep track of the best solution found. The results showed that decomposing a problem into subproblems

yields a better solution than solving the entire problem at once when bounded rational search is employed. Herrmann (2012) described a study in which different approaches for separating the Inventory Slack Routing Problem (a complex vehicle routing problem) were simulated. Again, a random search was used to simulate how a bounded rational human decision-maker would solve each subproblem. The results showed that the structure of the separation and the objectives used in each subproblem significantly affected the quality of the solutions that are generated. Additional details about the Inventory Slack Routing Problem can be found in the report by Montjoy and Herrmann (2012).

Hong and Page (2004) modeled problem-solvers of limited ability as heuristics; each heuristic searches a finite set of solutions until it cannot find a better one. Thus, the problem-solver is conducting a type of hill-climbing search. Hong and Page studied teams of such problem-solvers and identified conditions under which a diverse set of problem-solvers will likely perform better than a team of high-performing individuals. In their work, the problem-solvers searched a finite set of solutions (the size ranged from 200 to 10,000). A problem-solver was characterized by how many and which points near the current solution it would consider. Essentially, different problem-solvers had different neighborhood definitions.

The research described in the current paper models decision-makers like the problem-solvers that Hong and Page considered, and the search spaces used are similar to those that Hong and Page used. This research did not measure the value of diversity; instead it studied how separating the problem and allocating the team's resources affect the quality of the solutions that teams of problem-solvers generated.

Separations

A separation is a process that solves a sequence of subproblems. The concept of separation is similar (but not identical) to the idea of decomposition. Both replace a large design optimization problem with a set of subproblems. In a typical decomposition approach, a second-level problem must be solved to coordinate the subproblem solutions in an iterative manner.

Separation, on the other hand, does not require subsequent coordination. It is a decentralized and sequential approach in which a large problem is divided into subproblems. The solution to one subproblem will provide the inputs to one or more subsequent subproblems, but there is no higher-level problem to coordinate the solution. Note that the separation does not have to be a simple sequence of subproblems; it may have subproblems that are solved in parallel at places. A given separation specifies a partial order in which the subproblems are solved. A different order of subproblems would be a different separation and would lead to a different solution.

The subproblems' objective functions are surrogates for the original problem's objective function. These surrogates come from substituting simpler performance measures that are correlated with the original one, eliminating components that are not relevant to that subproblem, or from removing variables that will be determined in another subproblem.

Like dynamic programming, separation may solve a set of subproblems and use the solution of one problem to solve another. Typically dynamic programming recursively solves a set of subproblems (corresponding to a set of possible states) starting with a trivial subproblem. By contrast, separation does not contain this special recursive structure; therefore, solving a subproblem considers only the decisions that have been made.

Also, despite the similar name, separation is not the same as separable programming, a branch of mathematical programming that concerns nonlinear optimization problems in which the objective function and the constraints are sums of single-variable functions (cf. Stefanov, 2001). Separable programming approaches use a linear program to approximate the original problem and employ a type of simplex algorithm to find a solution. By contrast, separation replaces the original problem with a set of subproblems.

This study considered teams of bounded rational decision-makers (problem-solvers, or agents) and different approaches for solving various optimization problems.

In the basic problem, a problem-solver is given a set of points from 1 to n, and associated with each point i is a fixed positive value $V(i)$. The decision-maker wants to find the best point, that is, the one that has the largest value, but he is limited, however, to searching the space using a heuristic.

The points are arranged in a circular list. From the current point, a heuristic checks K positions that lie within L points down the list on the circle. A problem-solver accepts the new point if it has a better value than the current point. For the next check (regardless of whether the previous point was accepted) the problem-solver uses the next increment in the heuristic (or begins again with the first one). For example, if the heuristic has three positions (2, 3, and 10), the problem-solver first checks the point that is 2 positions down from the current point, accepts the new point as the current point if it is better, then checks the point that is 3 positions down from the current point, accepts that new point as the current point if it is better, then checks the point that is 10 positions down from the current point, accepts that new point as the current point if it is better, and so on until none of three positions has a better value.

This study considered the following problems and their separations.

In problem P1, n points are organized into N sets, and every set has n/N points. Every set s has a value $W(s)$, and the value of a point i in set s is $V(i) = W(s) + X(i)$. Thus, the values of the points are correlated with the set.

A team of two agents can solve this problem using two approaches. In the "all-at-once" approach, the first agent searches the entire space of n points until he stops at a locally optimal solution, and then the second agent begins searching from that point until he stops at a locally optimal solution (because the two agents use different heuristics, a point that is locally optimal for the first agent may not be locally optimal for the second agent).

The agents can also separate the problem into two subproblems that are solved sequentially. The first subproblem is to find the best set, and the second subproblem is to find the best point within that set. The first agent searches the list of N sets until he stops at a locally optimal set, and then the second agent searches that set of points until he stops at a locally optimal solution.

The "value" of a set was determined, in some cases, by calculating the largest value of the points in the set and, in other cases, by calculating the average value of the points in the set. In the first case, of course, the point with the largest value is always in the set with the largest value. In the second case, it may happen that the point with the largest value is not in the set with the largest value.

In problem P2, there were two sets of "components," and the points are the combinations of these two components. In particular, let j be a member of the first set of components and let $X(j)$ be its value. Let k be a member of the second set of components and let $Y(k)$ be its value. For every combination of j and k there is a point i, and the value of a point i is $V(i) = X(j) + Y(k)$.

A team of two agents can solve this problem using two approaches. The first approach is the "all-at-once" approach. The agents can also separate the problem into two subproblems that are solved in parallel. The first subproblem is to find the best component in the first set, and the second subproblem is to find the best component in the second set. The combination of these two components is the result. The first agent searches the first set of components while the second agent searches the second set of components.

Problem P3 is similar to problem P2. There were three sets of "components," and the points are the combinations of these three components. In particular, let j be a member of the first set of components and let $X(j)$ be its value. Let k be a member of the second set of components and let $Y(k)$ be its value. Let l be a member of the third set of components and let $Z(l)$ be its value. For every combination of j and k and l there is a point i, and the value of a point i is $V(i) = X(j) + Y(k) + Z(l)$.

A team of three agents can solve this problem using two approaches. The first approach is the "all-at-once" approach (the third agent begins his search where the second agent stops). The agents can also separate the problem into three subproblems that are solved in parallel. The first subproblem is to find the best component in the first set, the second subproblem is to find the best component in the second set, and the third subproblem is to find the best component in the third set. The combination of these three components is the result. The first agent searches the first set of components while the second agent searches the second set of components and the third agent searches the third set of components.

In problem D1, there were two sets of components. The first component represented a "design," and the second component represented a "price." The designs were grouped into subsets that represent "concepts."

Every concept c has a value $W(c)$, and the value of a design j that is in that subset is $V(j) = W(c) + X(j)$. Let $P(p)$ be the amount of the p-th price. Associated with every combination (j, p) of a design j and price p is its "profit," which equals $f(V(j), P(p))$, where

$$f(v,z) = \frac{e^{v-z}}{1+e^{v-z}}\left(z - \tfrac{1}{8}v^2\right).$$

The first part of this expression represents the relative demand (sales) for the product, and the second part is the profit per unit of the product sold. In problems P2 and P3, the desirability of one component did not depend upon the other(s) selected. In this problem, however, the desirability of a design depends upon the price selected and vice versa. That is, these two variables are coupled, so it would be ineffective to consider them independently.

The problem is to find the design and price combination that has the greatest profit. A team of three agents can solve this problem using various approaches. The first approach is the "all-at-once" approach over the entire space of design-price combinations.

The agents can also separate the problem into three subproblems that are solved sequentially: The first subproblem is to find the best concept (the one with the greatest average value $\bar{V}$), the second subproblem is to find the best design (the one with the greatest value $V(j)$) that corresponds to that concept (is in that particular subset of designs), and the third subproblem is to find the best price, which is the one that, when combined with the best design found, yields the greatest profit. The first agent searches the set of concepts, then the second agent searches the associated subset of designs, and finally the third agent searches the set of prices. Let "CDP" denote this separation.

The agents can also separate the problem into two subproblems that are solved sequentially: The first subproblem is to find the best concept-price combination (the one with the greatest projected profit $f\left(\bar{V}, P(p)\right)$, and the second subproblem is to find the best

corresponding design (the one with the greatest value). The first agent searches the combinations of concepts and prices, and then the second agent searches the associated subset of designs. The third agent can help one of two agents by continuing that search. Let "CV" denote this separation.

A third separation was also considered. This is similar to the previous one, but the second subproblem is to find the corresponding design that has the greatest profit (in combination with the price found by solving the first subproblem). Let "CP" denote this separation.

Instances

To compare the performance of different separations, we randomly generated instances according to the following scheme.

For problem P1, we generated instances with n = 400, 1600, and 10,000. The number of sets depended upon n as follows: with n = 400, N = 10, 20, and 40; with n = 1600, N = 20, 40, and 80; and with n = 10,000, N = 50, 100, and 200.

The values of $W(s)$ were randomly drawn according to a uniform distribution on the interval [0, 100], and the values of $X(i)$ were randomly drawn according to a uniform distribution on the interval [0, B], where B = 10, 20, and 30.

When n = 400, we generated 50 instances for each combination of values for N and B. When n = 1600 and 10,000, we generated 10 instances for each combination of values for N and B.

We checked to determine how often the point with the largest value was indeed in the set with the largest value when the "value" of a set was determined by calculating the average value of the points in the set. Table 1 lists the results.

Table 1. The number of instances in which the point with the largest value was indeed in the set with the largest value when the "value" of a set was determined by calculating the average value of the points in the set. For n = 400, there were 50 instances for each combination of values for N and B. For n = 1600 and 10,000, there were 10 instances for each combination of values for N and B. (n is the total number of points, N is the number of sets, and B is the upper bound on the range for $X(i)$.)

n	N	B = 10	B = 20	B = 30
400	10	48	48	48
	20	48	44	39
	40	43	38	34
1600	20	10	10	9
	40	8	6	5
	80	8	6	5
10,000	50	10	9	8
	100	10	8	5
	200	6	4	2

For problem P2, let N_1 be the number of components in the first set, and let N_2 be the number of components in the second set. We generated 10 instances for each of the following combinations of N_1 and N_2: (10, 40), (20, 20), (20, 80), (40, 40), (50, 200), and (100, 100). This yielded a total of 60 instances with $n = N_1 N_2$ = 400, 1600, and 10,000 points. The values of $X(j)$ and $Y(k)$ were randomly drawn according to a uniform distribution on the interval [0, 100].

For problem P3, let N_1 be the number of components in the first set, let N_2 be the number of components in the second set, and let N_3 be the number of components in the third set. We generated 10 instances for each of the following combinations of (N_1, N_2, N_3): (20, 20, 20) and (40, 20, 10). This yielded a total of 20 instances with 8,000 points. The values of $X(j)$, $Y(k)$, and $Z(l)$ were randomly drawn according to a uniform distribution on the interval [0, 100].

For problem D1, let N_1 be the number of concepts, let N_2 be the number of design in each concept's subset, and let N_3 be the number of prices. We generated 10 instances for each of

the following combinations of (N_1, N_2, N_3): (10, 10, 80), (20, 20, 20), and (20, 40, 10). This yielded a total of 30 instances with 8,000 points. The values of $W(c)$ were randomly drawn according to a uniform distribution on the interval [0, 5]. The values of $X(j)$ were randomly drawn according to a uniform distribution on the interval [0, 1]. The prices $P(p)$ were randomly drawn according to a uniform distribution on the interval [0, 10]. Note that the optimal combination of design value and price is (4, 4), at which point the profit equals 1.

We also generated 30 instances of problem D1 in which the values of $W(c)$ were randomly drawn according to a uniform distribution on the interval [0, 3] and the values of $X(j)$ were randomly drawn according to a uniform distribution on the interval [0, 3].

Heuristics

A set of heuristics is described by the parameters L and K. We used the following eight combinations of (L, K): (6, 2), (6, 3), (12, 2), (12, 3), (12, 7), (20, 2), (20, 3), and (20, 7). For five of these combinations (all except those with $K = 7$) we considered every possible heuristic. (Because the sequence of the positions matters, the total number of different heuristics is $L \times (L-1) \times \cdots \times (L-K+1)$.) For the two combinations with $K = 7$ we randomly generated 2,000 distinct heuristics from among those possible because evaluating every heuristic would have required excessive computational effort. (For $L = 12$ and $K = 7$, there are 3,991,680 different heuristics; for $L = 20$ and $K = 7$, there are 390,700,800 different heuristics.)

Computational Experiments

The purpose of the computational experiments was to compare the performance of the separations. All of the algorithms were implemented in Matlab and executed using Matlab R2006b on a Dell Optiplex GX745 with Intel Core2Duo CPU 6600 @ 2.40 GHz and 2.00 GB

RAM running Microsoft Windows XP Professional Version 2002 Service Pack 3. The computational effort of the testing increased as the number of heuristics considered grew, but that is not an important performance metric in this work.

For a given K and L, teams of agents were created as follows. First, we randomly generated two permutations of the list of heuristics. Then, the first team included an agent using the first heuristic on the original list, an agent using the first heuristic in the first permutation, and an agent using the first heuristic in the second permutation. The next team included an agent using the second heuristic on the original list, an agent using the second heuristic in the first permutation, and an agent using the second heuristic in the second permutation. This continued until we had as many teams as heuristics.

For problems P1, P2, and P3, every team was run on every instance. Because an agent must start somewhere, the team's performance was determined by averaging its performance over every possible initial point. Two performance measures were considered: the average relative value of the point that the team returned (the relative value equals the value of the point divided by the best value for that instance) and the average number of points that had to be evaluated (including the initial point) during the search. This was averaged over all of the instances to get the average performance. The performance for particular values of K and L (a list of heuristics) was the average over all of the corresponding teams.

When the teams were searching the complete set of points in the all-at-once approach, the points were first "shuffled" by creating a random permutation and searching that list. This avoided any correlation of the points' values caused by the way in which the random values were generated. We tested using five different random permutations of every instance, but the performance was the same for all of the permutations (for any instance and list of heuristics).

For the all-at-once approaches that were solved by teams of two or three agents, we recorded the values of the solutions found by each agent (at its stopping point) in order to determine the incremental value that the additional agents had.

Thus, for every problem, problem size, and combination of *K* and *L*, we generated six values of performance: the average relative value (ARV) and average number of evaluations (ANE) of the all-at-once approach (for the first agent and the team) and the separation.

For Problem D1, we randomly selected 200 teams for each combination of *K* and *L*; each team had three agents that used three different heuristics. For this problem, we considered L = 12 and K = 2, 3, and 7. The agents were allocated to the separations' subproblems as follows: for the CDP separation, each agent solved one subproblem; for the CV separation, the first agent always solved the first subproblem, and the second agent always solved the second subproblem, and the third agent remained idle (the "C1V1" option), solved the first subproblem (the "C2V1" option), or solved the second subproblem (the "C1V2" option); and for the CP separation, the first agent always solved the first subproblem, and the second agent always solved the second subproblem, and the third agent remained idle (the "C1P1" option), solved the first subproblem (the "C2P1" option), or solved the second subproblem (the "C1P2" option).

We randomly generated 20 instances in each of the combinations of (N_1, N_2, N_3). We randomly selected 10 combinations of initial points; each combination includes a value in the set $\{1, \ldots, N_1\}$, a value in the set $\{1, \ldots, N_2\}$, and a value in the set $\{1, \ldots, N_3\}$. Thus, for each of the combinations of (N_1, N_2, N_3) and each combination of *K* and *L* and each solution approach, we generated 20,000 solutions, one for each team, instance, and combination of initial points. We determined the ARV and ANE across all teams, instances, and combinations of initial points.

To gain some insight into the relative performance of the solution approaches, we also determined, for a particular team, instance, and combination of initial points, whether the solution generated by each solution approach was a "competitive" solution (that is, its profit was within 1% of the best profit found by the solution approaches). These were totaled over the 20,000 combinations of teams, instances, and combinations of initial points. To gain some insight into the absolute performance of the solution approaches, we also determined, for a particular team, instance, and combination of initial points, whether the solution generated by each solution approach was a "quality" solution (that is, its profit was within 1% of the best possible profit for that instance).

Results

For problems P1, P2, and P3, we evaluated the performance of three solution approaches (the first agent solving the all-at-once problem, the two agents solving the all-at-once problem, and the separation approach) for each list of heuristics by the average relative value (ARV) and the average number of evaluations (ANE) across all instances of the same size, all heuristics, and all initial points.

For problem P1, Tables 2, 3, and 4 list the results for the first version of the problem with $n = 400$; the results for larger problem sizes are very similar. In general, the ARV and ANE increase as the heuristic size K increases. A heuristic with a larger value of K evaluates more points in the neighborhood of the current solution and is therefore more likely to find a point with a better value. Increasing L while K remains the same does not generally increase ARV and ANE. As B increases, the range of the point values increases, and the ARV decreases; in general, the decrease for the all-at-once approaches is greater than the decrease for the separation approach. The number of sets also influences the results; for $n = 400$, the ARV decreases more

when *N* (the number of sets) increases from 10 to 20 than it decreases when *N* increases from 20 to 40.

The method used to determine the performance of a set made no difference to the results.

The separation approach, on average, evaluates more points than the all-at-once approaches. The separation approach finds better solutions than the all-at-once approach with only one agent and usually (but not always) finds better solutions than the all-at-once approach with two agents.

In general, we see that the solution approach and the size of the heuristic influence the results the most.

Table 2. The results for $n = 400$ and $N = 10$. There were 50 instances for each combination of values for *N* and *B*. (*n* is the total number of points, *N* is the number of sets, *B* is the upper bound on the range for $X(i)$, and *L* and *K* describe the list of heuristics. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.*)

				All-at-once (one agent)		All-at-once (two agents)		Separation (two agents)	
n	*N*	*B*	(*L*, *K*)	ARV	ANE	ARV	ANE	ARV	ANE
400	10	10	(6, 2)	0.840	4.5	0.876	7.9	0.890	8.9
			(6, 3)	0.878	6.3	0.906	10.8	0.933	12.3
			(12, 2)	0.841	4.5	0.889	8.0	0.874	8.8
			(12, 3)	0.880	6.3	0.916	11.0	0.917	12.2
			(12, 7)	0.937	13.3	0.951	22.3	0.978	25.6
			(20, 2)	0.840	4.5	0.892	8.1	0.865	8.8
			(20, 3)	0.879	6.3	0.920	11.0	0.909	12.2
			(20, 7)	0.938	13.4	0.957	22.7	0.972	25.6
400	10	20	(6, 2)	0.817	4.5	0.852	7.8	0.883	8.9
			(6, 3)	0.854	6.2	0.882	10.7	0.927	12.3
			(12, 2)	0.818	4.5	0.864	8.0	0.869	8.8
			(12, 3)	0.856	6.2	0.893	10.9	0.913	12.3
			(12, 7)	0.916	13.3	0.932	22.3	0.976	25.5
			(20, 2)	0.818	4.5	0.868	8.1	0.861	8.8
			(20, 3)	0.856	6.3	0.897	11.0	0.905	12.2
			(20, 7)	0.917	13.4	0.939	22.7	0.970	25.6
400	10	30	(6, 2)	0.800	4.5	0.834	7.9	0.878	8.9
			(6, 3)	0.837	6.2	0.865	10.7	0.923	12.3
			(12, 2)	0.801	4.5	0.847	8.0	0.865	8.8
			(12, 3)	0.838	6.2	0.876	10.9	0.910	12.3
			(12, 7)	0.901	13.3	0.918	22.3	0.974	25.5
			(20, 2)	0.800	4.5	0.850	8.1	0.857	8.8
			(20, 3)	0.838	6.3	0.880	11.0	0.902	12.2
			(20, 7)	0.901	13.3	0.925	22.7	0.968	25.6

Table 3. The results for $n = 400$ and $N = 20$. There were 50 instances for each combination of values for N and B. (n is the total number of points, N is the number of sets, B is the upper bound on the range for $X(i)$, and L and K describe the list of heuristics. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.*)

				All-at-once (one agent)		All-at-once (two agents)		Separation (two agents)	
n	N	B	(L, K)	ARV	ANE	ARV	ANE	ARV	ANE
400	20	10	(6, 2)	0.815	4.5	0.851	7.9	0.857	9.0
			(6, 3)	0.853	6.2	0.883	10.8	0.895	12.5
			(12, 2)	0.816	4.5	0.864	8.0	0.851	8.9
			(12, 3)	0.855	6.3	0.893	10.9	0.893	12.4
			(12, 7)	0.916	13.3	0.932	22.2	0.958	26.0
			(20, 2)	0.816	4.5	0.867	8.0	0.843	8.8
			(20, 3)	0.855	6.2	0.896	11.0	0.887	12.3
			(20, 7)	0.916	13.3	0.937	22.6	0.956	26.0
400	20	20	(6, 2)	0.796	4.5	0.831	7.8	0.855	9.0
			(6, 3)	0.833	6.2	0.862	10.7	0.894	12.5
			(12, 2)	0.797	4.5	0.843	8.0	0.850	8.9
			(12, 3)	0.834	6.2	0.872	10.9	0.892	12.4
			(12, 7)	0.896	13.2	0.913	22.2	0.958	26.0
			(20, 2)	0.797	4.5	0.847	8.0	0.842	8.8
			(20, 3)	0.835	6.2	0.876	11.0	0.886	12.3
			(20, 7)	0.897	13.3	0.919	22.5	0.955	26.0
400	20	30	(6, 2)	0.783	4.5	0.817	7.9	0.855	9.0
			(6, 3)	0.820	6.2	0.849	10.7	0.895	12.5
			(12, 2)	0.784	4.5	0.829	8.0	0.849	8.9
			(12, 3)	0.820	6.2	0.858	10.9	0.892	12.4
			(12, 7)	0.883	13.2	0.901	22.2	0.958	26.0
			(20, 2)	0.784	4.5	0.833	8.0	0.841	8.8
			(20, 3)	0.821	6.2	0.862	11.0	0.885	12.3
			(20, 7)	0.883	13.2	0.908	22.5	0.956	26.0

Table 4. The results for n = 400 and N = 40. There were 50 instances for each combination of values for N and B. (n is the total number of points, N is the number of sets, B is the upper bound on the range for $X(i)$, and L and K describe the list of heuristics. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.*)

				All-at-once (one agent)		All-at-once (two agents)		Separation (two agents)	
n	N	B	(L, K)	ARV	ANE	ARV	ANE	ARV	ANE
400	40	10	(6, 2)	0.809	4.5	0.846	7.9	0.841	8.8
			(6, 3)	0.848	6.2	0.878	10.7	0.883	12.2
			(12, 2)	0.809	4.5	0.859	8.0	0.837	8.8
			(12, 3)	0.849	6.2	0.889	10.9	0.882	12.2
			(12, 7)	0.912	13.3	0.929	22.2	0.951	25.8
			(20, 2)	0.809	4.5	0.863	8.0	0.836	8.8
			(20, 3)	0.849	6.2	0.892	11.0	0.881	12.2
			(20, 7)	0.912	13.3	0.935	22.7	0.951	25.4
400	40	20	(6, 2)	0.793	4.5	0.829	7.8	0.843	8.8
			(6, 3)	0.831	6.2	0.861	10.7	0.886	12.2
			(12, 2)	0.793	4.5	0.842	8.0	0.837	8.8
			(12, 3)	0.832	6.2	0.871	10.9	0.882	12.2
			(12, 7)	0.895	13.2	0.912	22.1	0.952	25.7
			(20, 2)	0.793	4.5	0.845	8.0	0.836	8.8
			(20, 3)	0.832	6.2	0.875	11.0	0.882	12.2
			(20, 7)	0.896	13.3	0.920	22.7	0.953	25.4
400	40	30	(6, 2)	0.781	4.5	0.816	7.8	0.843	8.8
			(6, 3)	0.818	6.2	0.847	10.7	0.888	12.3
			(12, 2)	0.780	4.5	0.828	8.0	0.837	8.8
			(12, 3)	0.819	6.2	0.858	10.9	0.882	12.2
			(12, 7)	0.883	13.1	0.901	22.0	0.953	25.8
			(20, 2)	0.780	4.5	0.832	8.0	0.836	8.8
			(20, 3)	0.819	6.2	0.862	11.0	0.882	12.2
			(20, 7)	0.884	13.3	0.910	22.6	0.953	25.4

Tables 5 and 6 list the results for problem P2. (Table 6 shows the results for N_1 = 40 and N_2 = 40; the results for other values of N_1 and N_2 were very similar.) Again, the ARV and ANE increase as the heuristic size K increases. A heuristic with a larger value of K evaluates more points in the neighborhood of the current solution and is therefore more likely to find a point with a better value. Increasing L while K remains the same does not generally increase ARV and ANE.

The separation approach, on average, evaluates more points than the all-at-once approaches. The separation approach finds better solutions than the all-at-once approach with

only one agent and, on average, finds better solutions than the all-at-once approach with two agents.

Changing N_1 and N_2, the number of items in each set of components, affects the ARV and ANE much less than changing *K*, the size the heuristics.

We found no correlation between the average value of the points returned in the all-at-once approaches and the average value of the points returned using the separation. That is, the heuristics that performed best in the separation approach did not necessarily perform the best in the all-at-once approaches.

Table 5. The results for Problem P2. (N_1 and N_2 are the number of components in each set, and L and K describe the list of heuristics. ARV = *Average Relative Value.*)

			All-at-once (one agent)	All-at-once (two agents)	Separation (two agents)
N_1	N_2	(L, K)	ARV	ARV	ARV
10	40	(6, 2)	0.778	0.811	0.865
		(6, 3)	0.813	0.841	0.907
		(12, 2)	0.777	0.822	0.857
		(12, 3)	0.813	0.851	0.900
		(12, 7)	0.876	0.894	0.963
		(20, 2)	0.777	0.826	0.855
		(20, 3)	0.814	0.856	0.899
		(20, 7)	0.879	0.905	0.963
20	20	(6, 2)	0.754	0.789	0.853
		(6, 3)	0.791	0.821	0.900
		(12, 2)	0.756	0.803	0.844
		(12, 3)	0.794	0.834	0.890
		(12, 7)	0.861	0.880	0.964
		(20, 2)	0.756	0.806	0.838
		(20, 3)	0.793	0.838	0.885
		(20, 7)	0.863	0.892	0.959
20	80	(6, 2)	0.747	0.780	0.836
		(6, 3)	0.782	0.811	0.879
		(12, 2)	0.748	0.792	0.834
		(12, 3)	0.783	0.822	0.877
		(12, 7)	0.848	0.868	0.945
		(20, 2)	0.748	0.796	0.830
		(20, 3)	0.784	0.827	0.874
		(20, 7)	0.851	0.879	0.944
40	40	(6, 2)	0.759	0.793	0.852
		(6, 3)	0.795	0.824	0.894
		(12, 2)	0.760	0.805	0.850
		(12, 3)	0.796	0.835	0.892
		(12, 7)	0.861	0.880	0.953
		(20, 2)	0.760	0.809	0.847
		(20, 3)	0.797	0.839	0.891
		(20, 7)	0.863	0.889	0.954
50	200	(6, 2)	0.722	0.757	0.816
		(6, 3)	0.759	0.789	0.862
		(12, 2)	0.723	0.770	0.816
		(12, 3)	0.761	0.803	0.863
		(12, 7)	0.831	0.852	0.935
		(20, 2)	0.723	0.775	0.815
		(20, 3)	0.762	0.807	0.863
		(20, 7)	0.833	0.863	0.936
100	100	(6, 2)	0.731	0.764	0.818
		(6, 3)	0.766	0.795	0.861
		(12, 2)	0.731	0.777	0.819
		(12, 3)	0.768	0.807	0.864
		(12, 7)	0.834	0.854	0.932
		(20, 2)	0.731	0.781	0.819
		(20, 3)	0.768	0.812	0.863
		(20, 7)	0.836	0.836	0.932

Table 6. The results for Problem P2. (N_1 and N_2 are the number of components in each set, and *L* and *K* describe the list of heuristics. ANE = *Average Number of Evaluations.*)

			All-at-once (one agent)	All-at-once (two agents)	Separation (two agents)
N_1	N_2	(*L*, *K*)	ANE	ANE	ANE
40	40	(6, 2)	4.5	7.9	9.0
		(6, 3)	6.2	10.7	12.5
		(12, 2)	4.5	8.0	9.0
		(12, 3)	6.2	10.9	12.5
		(12, 7)	13.2	22.2	26.4
		(20, 2)	4.5	8.0	8.9
		(20, 3)	6.2	11.0	12.5
		(20, 7)	13.3	22.6	26.4

For problem P3, again, the ARV and ANE increase as the heuristic size *K* increases. Increasing *L* while *K* remains the same does not generally increase ARV and ANE. Table 7 presents the results for the all-at-once approaches (after the first agent, after the second agent, and after the third agent) and for the separation approach in which each agent searches one set of components. Compared to the all-at-once approaches, the separation approach, on average, evaluates more points (22.3% more) but finds better solutions (the average increase in ARV is 0.091).

When the number of components in each set changes, the ANE does not change, and the ARV increases. The increase in ARV due to this change is less than the increase in ARV due to increasing *K*.

Table 7. The results for the problem P3. (N_1, N_2, and N_3 are the number of components in the three sets, and *L* and *K* describe the list of heuristics. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.*)

				All-at-once (one agent)		All-at-once (2 agents)		All-at-once (3 agents)		Separation (3 agents)	
N_1	N_2	N_3	(*L*, *K*)	ARV	ANE	ARV	ANE	ARV	ANE	ARV	ANE
20	20	20	(6, 2)	0.711	4.5	0.740	7.8	0.756	11.1	0.845	13.3
			(6, 3)	0.741	6.2	0.767	10.7	0.777	14.9	0.891	18.5
			(12, 2)	0.711	4.5	0.750	8.0	0.770	11.3	0.843	13.3
			(12, 3)	0.743	6.2	0.778	10.9	0.795	15.3	0.889	18.5
			(12, 7)	0.802	13.2	0.821	22.2	0.828	30.6	0.960	39.5
			(20, 2)	0.711	4.5	0.754	8.0	0.777	11.4	0.835	13.3
			(20, 3)	0.743	6.2	0.781	11.0	0.801	15.5	0.883	18.5
			(20, 7)	0.804	13.3	0.830	22.6	0.842	31.3	0.957	39.3
40	20	10	(6, 2)	0.733	4.5	0.763	7.9	0.779	11.1	0.866	13.3
			(6, 3)	0.764	6.2	0.790	10.7	0.801	14.9	0.908	18.4
			(12, 2)	0.733	4.5	0.773	8.0	0.793	11.3	0.861	13.3
			(12, 3)	0.765	6.2	0.800	10.9	0.818	15.3	0.904	18.4
			(12, 7)	0.825	13.2	0.843	22.2	0.850	30.5	0.966	38.8
			(20, 2)	0.733	4.5	0.776	8.0	0.799	11.4	0.857	13.2
			(20, 3)	0.765	6.2	0.804	11.0	0.824	15.5	0.901	18.3
			(20, 7)	0.826	13.3	0.852	22.6	0.864	31.3	0.964	38.2

For problem D1, again, the ARV and ANE increase as the heuristic size *K* increases. Table 8 presents the results for the all-at-once approach (after the first agent, after the second agent, and after the third agent); Table 9 presents the results for the separations. The ARV of the third separation is less than the ARV of the agents' all-at-once search when *K* was 2 or 3 but is greater than the ARV of the agents' all-at-once search when *K* was 7. The ARV of the other two separations is less than the ARV of the third separation. The inferior performance of the second separation (compared to the third separation) demonstrates the importance of the objective function. In the second separation, the second agent seeks to maximize the value of the design without considering the price variable. In the third separation, the second agent seeks to maximize the profit of the design.

The ANE of the first separation (which involved three agents) was greater than the ANE of the other two separations and the ANE required by the three agents' all-at-once search.

Table 10 presents the results on how "competitive" each solution approach was. The number of competitive solutions generated by the third separation is less than the number of competitive solutions generated by the agents' all-at-once search when *K* was 2, about the same when *K* was 3, and is greater when *K* was 7. The all-at-once searches generated more competitive solutions as *K* increased, but the first separation generated fewer competitive solutions as *K* increased; although the quality of its solutions increased (as shown in Table 9), the quality of the solutions generated by the third separation increased more. The number of competitive solutions generated by the second separation decreased as *K* increased.

Table 8. The results for the all-at-once approach on problem D1. (N_1, N_2, and N_3 are the number of components in the three sets, and *L* and *K* describe the heuristics used by the agents. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.*)

				First agent		Second agent		Third agent	
N_1	N_2	N_3	(*L*, *K*)	ARV	ANE	ARV	ANE	ARV	ANE
10	10	80	(12, 2)	0.655	4.4	0.742	7.9	0.783	11.2
			(12, 3)	0.724	6.1	0.799	10.9	0.830	15.2
			(12, 7)	0.845	13.2	0.878	22.2	0.889	30.6
20	20	20	(12, 2)	0.637	4.5	0.729	8.0	0.775	11.4
			(12, 3)	0.710	6.3	0.792	11.0	0.824	15.4
			(12, 7)	0.841	13.4	0.877	22.4	0.888	30.7
20	40	10	(12, 2)	0.627	4.5	0.716	8.0	0.760	11.3
			(12, 3)	0.699	6.2	0.779	11.0	0.813	15.3
			(12, 7)	0.828	13.3	0.866	22.3	0.878	30.8

Table 9. The results for the problem D1. (N_1, N_2, and N_3 are the number of components in the three sets, and *L* and *K* describe the heuristics used by the agents. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.*)

				Separation					
				CDP		C1V1		C1P1	
N_1	N_2	N_3	(*L*, *K*)	ARV	ANE	ARV	ANE	ARV	ANE
10	10	80	(12, 2)	0.671	13.2	0.653	8.7	0.713	8.7
			(12, 3)	0.738	18.3	0.711	12.1	0.785	12.2
			(12, 7)	0.816	38.0	0.807	25.4	0.893	25.3
20	20	20	(12, 2)	0.683	13.3	0.649	8.8	0.719	8.9
			(12, 3)	0.749	18.4	0.713	12.3	0.791	12.4
			(12, 7)	0.817	38.6	0.816	25.9	0.902	26.1
20	40	10	(12, 2)	0.591	13.3	0.622	8.8	0.698	8.9
			(12, 3)	0.658	18.4	0.691	12.3	0.776	12.4
			(12, 7)	0.753	38.4	0.800	26.5	0.897	26.5

Table 10. The results for the problem D1. The number of solutions (out of 20,000) that were competitive. (N_1, N_2, and N_3 are the number of components in the three sets, *L* and *K* describe the heuristics, and AA1, AA2, AA3 = All-at-once with one agent, two agents, and three agents;)

N_1	N_2	N_3	(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1P1
10	10	80	(12, 2)	4,718	6,843	8,422	5,351	3,755	7,577
			(12, 3)	4,934	7,151	8,575	4,496	3,624	8,516
			(12, 7)	6,503	8,106	8,799	2,329	3,669	11,277
20	20	20	(12, 2)	4,772	6,839	8,442	5,930	3,655	7,301
			(12, 3)	5,008	7,214	8,547	5,284	3,592	8,286
			(12, 7)	6,546	8,183	8,916	2,708	3,593	11,552
20	40	10	(12, 2)	4,737	6,803	8,460	5,166	3,881	8,158
			(12, 3)	4,946	7,261	8,723	4,241	3,786	9,266
			(12, 7)	6,450	8,148	8,860	2,111	3,606	12,487

We also considered a limited set of experiments to determine the best way to allocate three agents. These were tested using the heuristics with *L* = 12 and over the ten trials (out of the twenty originally tested) on which the three agents' all-at-once search performed moderately. In particular, the five trials on the three agents' all-at-once search performed the best and the five trials on the three agents' all-at-once search performed the worst were excluded. This reduced some of the variability of the results. The results include the number of runs in which each approach generated "competitive" solutions (those within 1% of the best profit found by these approaches) and the number of runs in which each approach generated a "quality" solution (within 1% of the best profit in that trial). The CP approach generated the best solutions, as shown in Tables 11, 12, and13: it has the best ARV, and it generated a larger number of competitive solutions and a larger number of quality solutions. The ANE of the approaches depends mostly upon *K*, the size of the heuristics used, and the number of agents.

Table 11. The results for the problem D1 for limited trials on the set of instances with $N_1 = 10$, $N_2 = 10$, and $N_3 = 80$. $W(c) \in [0,5]$, $X(j) \in [0,1]$. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.* NCS = *Number of Competitive Solutions*; NQS = *Number of Quality Solutions*.

(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1V2	C2V1	C1P1	C1P2	C2P1
	ARV									
(12, 2)	0.677	0.761	0.801	0.669	0.685	0.683	0.755	0.738	0.748	0.813
(12, 3)	0.744	0.817	0.845	0.735	0.735	0.732	0.792	0.805	0.811	0.867
(12, 7)	0.859	0.888	0.897	0.803	0.822	0.819	0.844	0.910	0.912	0.934
	ANE									
(12, 2)	4.4	7.9	11.2	13.2	8.7	11.1	11.2	8.6	11.0	11.1
(12, 3)	6.1	10.8	15.2	18.3	12.1	15.6	15.8	12.0	15.5	15.6
(12, 7)	13.0	22.0	30.3	38.0	25.4	32.9	33.4	25.2	32.7	33.2
	NCS									
(12, 2)	2,182	3,085	3,762	1,843	1,649	1,733	2,106	3,412	3,944	4,524
(12, 3)	2,358	3,354	3,964	1,236	1,517	1,554	1,881	3,954	4,325	5,165
(12, 7)	3,206	3,967	4,294	273	1,426	1,419	1,596	5,465	5,589	6,374
	NQS									
(12, 2)	413	541	642	97	236	240	292	730	835	912
(12, 3)	528	701	837	68	239	245	284	964	1,084	1,208
(12, 7)	928	1,153	1,256	7	273	284	289	1,727	1,828	1,985

Table 12. The results for the problem D1 for limited trials on the set of instances with $N_1 = 20$, $N_2 = 20$, and $N_3 = 20$. $W(c) \in [0,5]$, $X(j) \in [0,1]$. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.* NCS = *Number of Competitive Solutions*; NQS = *Number of Quality Solutions*.

(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1V2	C2V1	C1P1	C1P2	C2P1
	ARV									
(12, 2)	0.656	0.751	0.797	0.699	0.677	0.674	0.760	0.755	0.767	0.833
(12, 3)	0.735	0.814	0.844	0.761	0.741	0.736	0.806	0.825	0.831	0.884
(12, 7)	0.860	0.893	0.902	0.830	0.833	0.830	0.858	0.920	0.922	0.937
	ANE									
(12, 2)	4.5	8.0	11.4	13.4	8.7	11.2	11.2	8.9	11.4	11.3
(12, 3)	6.3	11.0	15.4	18.5	12.1	15.7	15.8	12.3	15.9	15.9
(12, 7)	13.3	22.2	30.5	38.6	25.6	33.4	33.4	25.9	33.8	33.8
	NCS									
(12, 2)	1,930	2,784	3,461	2,519	1,412	1,535	1,987	3,180	3,761	4,405
(12, 3)	2,140	3,030	3,543	2,254	1,458	1,511	1,902	3,797	4,219	5,033
(12, 7)	3,005	3,665	3,958	1,261	1,655	1,648	1,790	5,568	5,747	6,303
	NQS									
(12, 2)	356	516	656	492	310	313	413	902	1,091	1,223
(12, 3)	474	721	870	524	340	329	425	1,276	1,435	1,672
(12, 7)	979	1,224	1,324	229	563	572	564	2,529	2,657	2,848

Table 13. The results for the problem D1 for limited trials on the set of instances with $N_1 = 20$, $N_2 = 40$, and $N_3 = 10$. $W(c) \in [0,5]$, $X(j) \in [0,1]$. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.* NCS = *Number of Competitive Solutions*; NQS = *Number of Quality Solutions*.

(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1V2	C2V1	C1P1	C1P2	C2P1
	ARV									
(12, 2)	0.662	0.748	0.789	0.613	0.656	0.651	0.734	0.725	0.736	0.804
(12, 3)	0.734	0.811	0.842	0.676	0.721	0.717	0.785	0.800	0.807	0.863
(12, 7)	0.855	0.891	0.902	0.750	0.816	0.813	0.841	0.909	0.911	0.930
	ANE									
(12, 2)	4.5	8.0	11.3	13.3	8.9	11.4	11.4	8.8	11.4	11.3
(12, 3)	6.2	10.9	15.3	18.5	12.4	16.1	16.0	12.4	16.0	16.0
(12, 7)	13.3	22.2	30.6	38.2	26.6	34.5	34.4	26.5	34.3	34.4
	NCS									
(12, 2)	1,970	2,854	3,519	2,209	1,496	1,552	2,101	3,410	3,999	4,571
(12, 3)	2,151	3,128	3,709	1,654	1,552	1,551	2,074	4,030	4,505	5,276
(12, 7)	3,131	3,870	4,142	455	1,672	1,633	1,816	6,041	6,196	6,640
	NQS									
(12, 2)	343	514	636	457	381	350	524	1,054	1,241	1,384
(12, 3)	514	736	880	388	522	506	669	1,566	1,767	2,018
(12, 7)	1,080	1,327	1,429	68	811	810	911	2,994	3,083	3,351

For the instances of problem D1 in which the values of $W(c)$ were randomly drawn according to a uniform distribution on the interval [0, 3] and the values of $X(j)$ were randomly drawn according to a uniform distribution on the interval [0, 3], maximizing the value of the design performed even worse, as shown in Tables 14, 15, and 16. Using a third agent to extend the search for a large-value design after finding a concept-price combination led to less profitable solutions while using that third agent to extend the search for a better concept-price combination led to more profitable solutions. The separation in which the second agent searched for a large-value design generated solutions that were less profitable than those generated by the separation in which the first agent searched for the best concept, the second agent searched for a large-value design, and the third agent searched for the best price.

Using a third agent to extend the search for a profitable design after finding a concept-price combination led to more profitable solutions but using that third agent to extend the search for a better concept-price combination increased profitability even more.

Table 14. The results for the problem D1 for limited trials on the set of instances with $N_1 = 10$, $N_2 = 10$, and $N_3 = 80$. $W(c) \in [0,3]$, $X(j) \in [0,3]$. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.* NCS = *Number of Competitive Solutions*; NQS = *Number of Quality Solutions*.

(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1V2	C2V1	C1P1	C1P2	C2P1
	ARV									
(12, 2)	0.701	0.777	0.815	0.691	0.583	0.551	0.627	0.790	0.816	0.849
(12, 3)	0.762	0.827	0.853	0.752	0.580	0.544	0.600	0.858	0.872	0.902
(12, 7)	0.863	0.891	0.899	0.821	0.523	0.497	0.515	0.941	0.944	0.954
	ANE									
(12, 2)	4.3	7.8	11.0	13.2	8.8	11.2	11.3	8.7	11.2	11.2
(12, 3)	6.0	10.6	14.9	18.4	12.2	15.7	15.9	12.1	15.5	15.8
(12, 7)	12.7	21.5	29.9	38.2	25.7	33.2	33.7	25.4	33.0	33.4
	NCS									
(12, 2)	2,231	3,046	3,648	1,826	1,491	1,381	1,657	3,570	4,511	4,459
(12, 3)	2,336	3,180	3,648	1,289	1,092	1,007	1,121	4,179	4,997	5,262
(12, 7)	2,823	3,399	3,624	395	351	335	247	6,010	6,544	6,734
	NQS									
(12, 2)	416	613	742	254	304	286	338	906	1,155	1,110
(12, 3)	523	754	894	189	292	257	298	1,304	1,548	1,685
(12, 7)	968	1,218	1,325	14	149	158	97	2,711	2,964	3,126

Table 15. The results for the problem D1 for limited trials on the set of instances with $N_1 = 20$, $N_2 = 20$, and $N_3 = 20$. $W(c) \in [0,3]$, $X(j) \in [0,3]$. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.* NCS = *Number of Competitive Solutions*; NQS = *Number of Quality Solutions*.

(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1V2	C2V1	C1P1	C1P2	C2P1
	ARV									
(12, 2)	0.704	0.790	0.827	0.685	0.554	0.514	0.603	0.825	0.851	0.870
(12, 3)	0.773	0.838	0.864	0.743	0.555	0.508	0.571	0.882	0.896	0.912
(12, 7)	0.876	0.901	0.908	0.804	0.458	0.425	0.445	0.945	0.948	0.954
	ANE									
(12, 2)	4.5	8.0	11.3	13.3	8.7	11.1	11.2	8.8	11.3	11.3
(12, 3)	6.2	10.9	15.2	18.4	12.1	15.7	15.8	12.3	15.9	15.9
(12, 7)	13.1	22.0	30.3	38.8	26.0	33.8	34.0	26.1	33.8	33.9
	NCS									
(12, 2)	1,884	2,681	3,281	2,033	1,291	1,225	1,407	3,596	4,711	4,355
(12, 3)	1,971	2,811	3,264	1,618	939	803	898	4,240	5,244	4,981
(12, 7)	2,617	3,191	3,457	420	236	181	159	6,226	6,870	6,721
	NQS									
(12, 2)	316	492	612	395	429	390	421	977	1,305	1,145
(12, 3)	431	637	734	355	355	282	324	1,401	1,778	1,624
(12, 7)	787	955	1,044	53	133	109	86	2,862	3,262	3,111

Table 16. The results for the problem D1 for limited trials on the set of instances with $N_1 = 20$, $N_2 = 40$, and $N_3 = 10$. $W(c) \in [0,3]$, $X(j) \in [0,3]$. ARV = *Average Relative Value*; ANE = *Average Number of Evaluations.* NCS = *Number of Competitive Solutions*; NQS = *Number of Quality Solutions*.

(*L*, *K*)	AA1	AA2	AA3	CDP	C1V1	C1V2	C2V1	C1P1	C1P2	C2P1
	ARV									
(12, 2)	0.668	0.757	0.801	0.645	0.524	0.487	0.580	0.807	0.835	0.865
(12, 3)	0.743	0.820	0.851	0.698	0.541	0.498	0.570	0.878	0.893	0.916
(12, 7)	0.865	0.896	0.905	0.763	0.468	0.437	0.461	0.953	0.956	0.963
	ANE									
(12, 2)	4.5	8.0	11.3	13.3	8.8	11.4	11.3	8.9	11.4	11.3
(12, 3)	6.3	11.0	15.4	18.4	12.4	16.0	15.9	12.4	16.0	15.9
(12, 7)	13.3	22.2	30.6	38.1	26.2	34.2	34.1	25.8	33.8	33.6
	NCS									
(12, 2)	1,713	2,537	3,145	1,908	1,252	1,124	1,565	3,818	4,948	4,852
(12, 3)	1,884	2,738	3,259	1,295	1,035	847	1,167	4,662	5,659	5,704
(12, 7)	2,646	3,294	3,540	323	393	299	356	6,674	7,232	7,370
	NQS									
(12, 2)	319	586	814	347	285	220	383	1,088	1,424	1,435
(12, 3)	555	910	1,145	239	290	199	330	1,745	2,140	2,202
(12, 7)	1,290	1,704	1,861	16	129	99	99	3,696	4,111	4,222

Summary and Conclusions

This paper presented the results of simulating various separations of some simple optimization problems. The simulations are models of how bounded rational decision-makers make choices, and their purpose is to help assess the quality of different separations and the assignment of a team's resources to their subproblems.

The results indicate that teams can generally find better solutions by separating a large optimization problem into subproblem. This was true for problems that involve combining independent components and for problems that had coupled variables. Moreover, the allocation of effort affected the quality of the solutions found as well. In some cases, additional effort led to lower-quality solutions because the agents were optimizing the wrong objective.

Because human decision-makers have limitations and cannot optimize, well-designed separations are the best way to find quality solutions and make decisions. Trying to set everything all-at-once will lead to problems that are too large to solve well because the large

solution space makes it difficult for the bounded rational decision-maker to approach an optimal solution.

These results reinforce the conclusions of Herrmann (2010) about the usefulness of separating complex optimization problems for bounded rational decision-makers. They also demonstrate the usefulness of this simulation approach for comparing separations that have different subproblems and different allocations of team resources.

Acknowledgements

The author appreciates Ross Hammond's pointers to some important related work.

References

Gigerenzer, Gerd, Peter M. Todd, and the ABC Research Group, 1999, *Simple Heuristics That Make Us Smart*, Oxford University Press, New York.

Gurnani, Ashwin, and Kemper Lewis, 2008, "Collaborative, Decentralized Engineering Design at the Edge of Rationality," *Journal of Mechanical Design*, 130(12), 121101.

Herrmann, Jeffrey W., "Progressive Design Processes and Bounded Rational Designers," *Journal of Mechanical Design*, August 2010, Volume 132, Issue 8, 081005 (8 pages). doi:10.1115/1.4001902

Herrmann, Jeffrey W., "Separating the Inventory Slack Routing Problem," Technical Report TR_2012-8, Institute for Systems Research, University of Maryland, College Park, July, 2012. Available online at http://hdl.handle.net/1903/12865.

Hong, Lu, and Scott E. Page, "Groups of diverse problem solvers can outperform groups of high-ability problem solvers," *Proceedings of the National Academy of Sciences of the United States of America*, Volume 101, Number 46, pages 16385-16389, 2004.

March, James, and Herbert Simon, 1993, *Organizations*, second edition, Blackwell, Cambridge, Massachusetts.

Montjoy, Adam, and Jeffrey W. Herrmann, "Optimizing Urgent Material Delivery by Maximizing Inventory Slack," Technical Report 2012-6, Institute for Systems Research, University of Maryland, College Park, June 12, 2012. Available online at `http://hdl.handle.net/1903/12499`.

Nutt, Paul C., 2005, "Search During Decision-Making," *European Journal of Operational Research*, 160, pp. 851-876.

Simon, Herbert A., 1981, *The Sciences of the Artificial*, second edition, MIT Press, Cambridge, Massachusetts.

Simon, Herbert A., 1997a, *Models of Bounded Rationality*, Volume 3, The MIT Press, Cambridge, MA.

Simon, Herbert A., 1997b, *Administrative Behavior*, fourth edition, The Free Press, New York.

Stefanov, Stefan M., 2001, *Separable Programming*: Theory and Methods, Kluwer Academic Publishers, Dordrecht, Netherlands.

Wang, Y., and Chiew, V., 2008, "On the Cognitive Process of Human Problem Solving," *Cognitive Systems Research*, doi:10.1016/j.cogsys.2008.08.003.