Pragmatism and Spatial Layout Design

Susan L. Epstein[1], Bernard Moulin[2], Walid Chaker[2], Janice Glasgow[3], and Jeremi Gancet[2]

[1] Department of Computer Science, Hunter College and The Graduate School of The City University of New York, New York, NY 10021, USA
susan.epstein@hunter.cuny.edu
[2] Computer Science Department and Research Center of Geomatics, Laval University, Ste Foy, QC G1K 7P4, Canada
bernard.moulin@ift.ulaval.ca, walid.chaker@ift.ulaval.ca, jgancet@yahoo.com
[3] Department of Computing and Information Science, Queen's University, Kingston, Ontario, K7L3N6, Canada
Janice@cs.queensu.ca

Abstract. Design problems address multiple, ill-defined goals in a computationally intractable search space. Our approach to spatial layout design capitalizes on devices that people use to control search. We advocate multiple passes at increasing, class-based levels of granularity, with heavy infusions of knowledge. Our approach makes brief, inexpensive searches from good starting points, seeking ideas that satisfy multiple criteria simultaneously. The result is an autonomous program for two-dimensional layout design of recreational parks.

Keywords. spatial design, constraint-based reasoning, cognitive structure of spatial knowledge, social and cultural organization of space, structure of geographic information

1 Introduction

Although computer science has made inroads in many areas of human expertise, design has thus far been mostly restricted to computer-aided design, with good reason. Design has many of the classic characteristics of an artificial intelligence (*AI*) problem — its practitioners make extensive use of domain-specific knowledge; it requires symbolic reasoning with both qualitative and numeric methods; it is laden with inexact, missing, or ill-defined information; and the problems themselves lack clear, clean algorithmic solutions. Nonetheless, the thesis of this work is that it is possible for a program to formulate original, task-specific solutions to a design problem.

Our focus here is two-dimensional spatial layout design, and our investigative domain is recreational parks. In *Anatomy of a Park*, Molnar and Rutledge identify three phases of park construction: survey, analysis, and synthesis [1]. In the *survey*, the designer develops constraints and goals, seeking out purpose-directed reasoning as well as pre-existing resources and constraints. In the *analysis*, the designer extracts the relationships among the elements to be included, and in the *synthesis* a design is produced and then refined. Our work focuses only on synthesis; we assume that survey and

analysis have already been performed by able humans. Thus the problem is to provide a design that incorporates all the elements while abiding by the constraints and goals.

Our approach, *pragmatic design synthesis*, is a limitedly rational one to solution generation, an approach modeled on human designers [2-4]. For the two-dimensional layout design of a recreational park, pragmatic design synthesis capitalizes on the ways that people address the problem: object classes, levels of detail, and the infusion of knowledge. Our primary resources have been a successful empirical investigation for theoretical parks on an empty grid [5], and expert writings on architectural and landscape design for real-world data [1, 6-15]. The theory described here is the foundation for an ongoing, large-scale project that employs real-world data and provides urban and landscape architects with automated support for the design, or redesign, of parks.

The next section of this paper defines spatial layout design, introduces our running example, explains the challenges inherent in spatial layout design, and targets some devices that people use to solve these problems. The third section constitutes our theoretical approach to the design of recreational parks. The fourth section discusses the roles in design of geographic information, alternatives, and navigability, and includes implementation results.

2 The Problem

Informally, within some setting *spatial layout design* creates an arrangement of a set of partially-specified objects according to a set of objectives. The setting is called the *design frame*; it includes both the *site*, the geographical area in which the objects are to be arranged, and the *periphery*, the area immediately adjacent to the site. The design frame also includes any pre-existing objects deemed fixed, such as a body of water, a public utility location, or access to a transportation system. Fig. 1 begins our running example of a spatial layout design problem for a recreational park. The site includes a pond; the periphery includes two streets and two avenues.

Each object in spatial layout design has a prespecified description, its *property values*. These may be detailed and rigid (e.g., a tennis court whose dimensions are fixed and whose orientation to sun and wind are relevant) or more flexible (e.g., a sandbox). Spatial layout design is said to *anchor* an object within a site when it sizes, fixes, and orients the object according to its property values. A building specified only as 600 square units, for example, might be anchored by making it a 20×30 rectangle centered on a particular point on the site, and oriented so that its shorter side faces due north.

A problem in spatial layout design is specified by a design frame, a set of objects (e.g., the list in Fig. 1) with descriptions, and *criteria*, a set of specifications for the way the objects are to be anchored. The criteria are either required (*constraints*) or desired (*principles*). Constraints typically describe the objects themselves (e.g., size, shape, compass orientation) and their placement on the site, relative to the site's boundaries and relative to each other (e.g., a parking lot near the periphery, a restroom near a playground). Principles serve as metrics for good design; they are more vague than constraints, and are generally conditions to be optimized, rather than tested. Examples of principles include ease of access from a particular location to the periphery, and aesthetic pleasure during traversal.

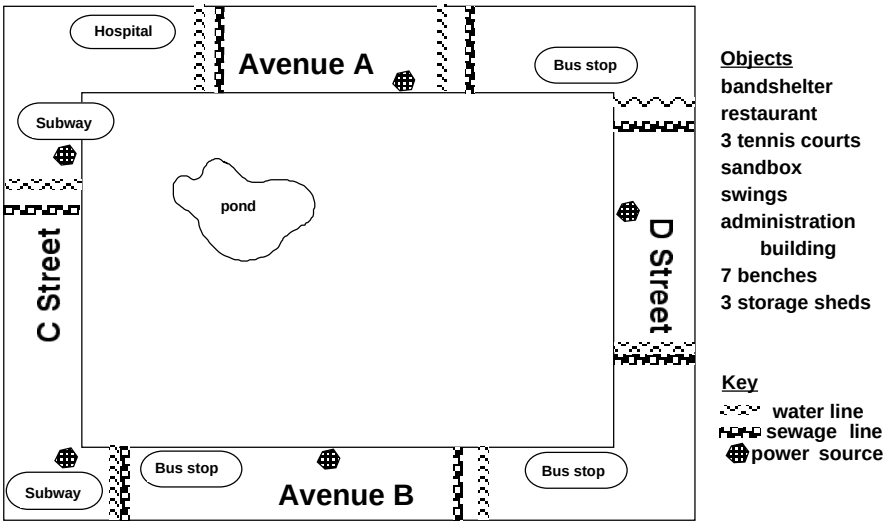

Fig.1. A design frame for an urban park with a list of objects to be included

A *solution* to a problem in spatial layout design anchors all the objects within the site, and satisfies all the constraints, without any of the objects overlapping. Thus *design synthesis* accepts as input a design frame, objects to be anchored within the site, and criteria that describe the way those objects may be anchored. Design synthesis produces as output a set of designs that anchor all the objects in accordance with the constraints and are highly-rated by the principles.

Problems in spatial layout design, along with its solution, can be treated as a *cases* (previously solved problems and their solutions) in a case-base of design experiences. Case-based reasoning in this domain can provide initial designs, adapt existing designs, or be used to evaluate possible solutions. Case-based reasoning has previously been applied to the problem of landscape design. Rather than focusing on the initial design layout, the CYCLOPS system [16] combines constraint-based solution generation with case-based reasoning to repair landscape architecture designs.

2.1 The Issues

Design problems are hard because, in very large search spaces, they aspire to multiple ill-defined goals [2]. A solution to a design problem is expected to satisfy both specific, real-world constraints (e.g., size, cost) and less well-defined, but nonetheless relevant specifications (e.g., aesthetic properties). This section explores the difficulties design problems present within the traditional AI approach.

Consider a design synthesis problem that specifies *n* objects to be anchored within an otherwise empty site. Each object is already sized and oriented; one need only locate the objects with respect to the single constraint that they do not overlap. In this event, anchoring need only specify a *location*, the coordinates for the center of each object. A design synthesis state could then be described as a vector of *n* elements, each of which

is a location or "unknown." If any entry in the vector is "unknown," the state is said to be *partial*; otherwise it is *complete*. If the design described by the vector abides by the no-overlap constraint, the state is said to be *legal*; otherwise it is *illegal*. A solution is simply a legal, complete state.

The traditional AI approach to design synthesis would be to place one object at a time on the grid in a way that satisfies the constraints, until all objects are placed. If objects remain to be placed, but no unplaced object has a placement that does not overlap an already-placed object, the program would backtrack to some previous choice point, change that decision, and retract all the decisions that had followed it. This can be characterized as search for a solution through a space of legal, partial states.

Unfortunately, the number of possible anchorings is enormous, even for a simple problem of this kind. One particular problem with only 13 objects had about 1.3×10^{26} possible complete states for those objects on a small grid, the vast majority of which were illegal [5]. Thus design synthesis appears to require search through an unmanageably large space. Equally problematic is that, although enlarging the site size may simplify a design problem for people, it makes it more difficult for this kind of search — each object then has more possible locations, which increases the branching factor.

As described thus far, state-space search has also ignored a hallmark of design that distinguishes it from many other AI problems: its insistence upon multiple goals. The 13-object problem, for example, included 14 fairly simple constraints, such as "this object must be tangent to the northern boundary of the site" or "this object must have its center within 80% of the site's center." Additional constraints eliminate previously-legal complete states, so that solutions in the state space are even more sparse.

Recall that, in addition to its required constraints, the design problem has a set of desired principles. The correct, yet computationally intractable, approach for a solution that optimizes the principles is to examine every solution and select the one the principles rate highest. A program cannot afford to neglect these principles, for they include the aesthetics that have thus far made design the province of people. Principles, however, tend to be vague and computationally costly [3]. Yet, somehow, human designers produce satisfactory solutions.

2.2 Human Techniques

The approach we advocate capitalizes on practical techniques human designers use to control search: object classes and granularity, knowledge infusion, and agent-based simulation. Human layout design experts typically assign specified objects to *classes*, sets of functionally and/or structurally similar objects [3]. Thus the sandbox and the swings of Fig. 1 would be in the playground class, the restaurant and the band shelter in the entertainment class. A particularly important object, such as the administration building, could form its own class. This perspective transforms the list of objects in Fig. 1 into the classification of Table 1, which groups objects by the activity they are intended to support. Here a *use area* is a section of the park intended for a particular activity, such as a playground or an entertainment area. All objects in a use area must belong to the same class, but there may be more than one use area with the same class of objects, for example, there may be two playgrounds in a large park.

Table 1. Object classes for the problem of Fig. 1

Use area	Type	Objects
Maintenance	major	Administration building
Entertainment	major	Band shelter, restaurant
Playground	major	Sandbox, swings
Playground	minor	Bench 1, bench 2, bench 3
Playing fields	major	Tennis court 1, tennis court 2, tennis court 3
Playing fields	minor	Bench 4, bench 5
any	minor	Bench 6, bench 7
any	minor	Shed 1, shed 2, shed 3

Classification permits a designer to focus either on fewer objects at a time (those within a class) or, as we shall see, only on the use areas themselves [1]. Indeed, human experts produce designs of increasing *granularity* (level of detail). Their early designs focus on key objects or classes of objects, while subsequent efforts embellish the early work with details.

1.3 Computational Techniques

In general, design problems in AI are defined by constraints, and the problem solver is expected to provide a solution that satisfies them. The constraints may either *underspecify* the problem, so that there are many possible solutions, or *overspecify* the problem, so that no set of anchorings satisfies all the constraints. To date, most automated design systems have supported human designers, rather than initiated designs of their own. For example, ARCHIE [17] helps architects carry out conceptual design, and AskJef [18] assists human-computer interface designers.

A major result in AI is that properly-formulated knowledge expedites and refines search. Human experts rely heavily both on compendia of information (such as the 154-page soil treatise [19]) and on cases. The 13-object problem is difficult, in part because its site is an empty grid that offers no further restrictions. In the real world, sites provide extensive data, data that can be used to guide search. Section 4 further discusses the role of different knowledge sources in spatial layout design.

3 Pragmatic Design Synthesis

Pragmatic design synthesis uses the reformulation of Table 1 to execute three passes, from sketch to plan and finally to design. Each pass works with a different level of granularity, and each involves multiple, resource-limited searches.

3.1 The Sketch

The first step in the design process, at the coarsest level of granularity, positions use areas. A successful search in this pass ends in a *sketch*. Kinds of use areas include playgrounds, sports fields, water-related activities, entertainment, forests, meadows, food service, maintenance facilities, and container network skeletons. A *container network* is a connected graph intended to transport substances or objects. Examples of a container network include roads, power lines, and hiking trails. At its coarsest level, in the sketch, a container network is represented by a container network *skeleton* that gives the fundamental but unelaborated shape of the container network to be used in the sketch. For example, Frederick Law Olmstead, a famed designer of urban parks, favored an oval about three-fourths the size of the site for his road skeletons.

The specification of the typical use area stipulates its left-right (longest embeddable horizontal line in the plane) and top-bottom dimensions (longest embeddable vertical line in the plane). It may also specify expected traffic (source and destination with respect to other use areas and the periphery, including estimated volume and frequency of access), and tolerances for deformation and rotation. Tolerance for deformation indicates the manner and the degree to which the boundary of the oval may be stretched or contracted. Each use area also has a refinement function, explained in Section 4.1. Container networks require special treatment — because their construction is computationally difficult, pragmatic design synthesis includes a case library of container network skeletons. Use area constraints frequently describe distances between pairs of use areas. One might, for example, specify that a picnic area be near a pond, that a playground be near a bus stop, or that the maintenance area be near the entertainment area but far from the picnic area.

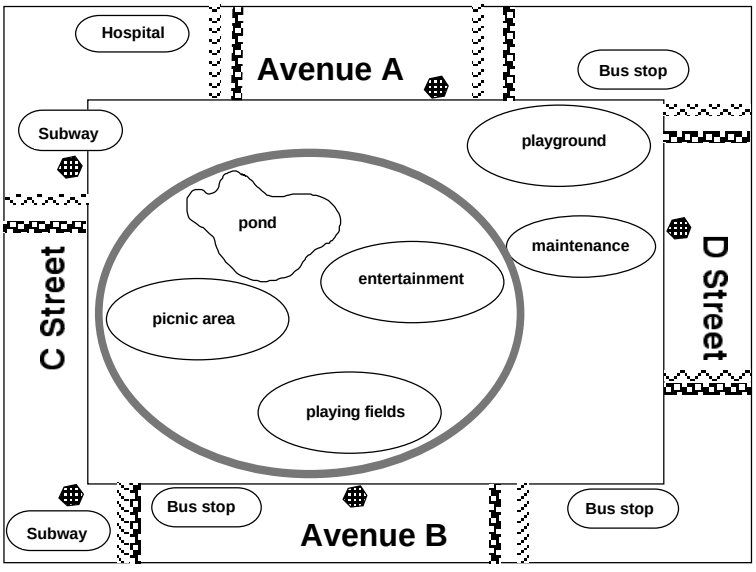

Fig. 2. A sketch for the design frame of Fig. 1.

A sketch anchors each use area in the site. A use area appears as a labeled oval whose area approximates its expected *footprint* (projection onto the plane of the site) in the final design. For the use areas of Table 1 and the design frame of Fig. 1, the sketch in Fig. 2 might be generated

3.2 The Plan

Next, within each use area, the second pass positions major items. A successful search of this kind ends in a *use area plan.* Detailing the best sketches with the best use area plans for each of the use areas produces *plans.* Each use area contains any number of objects that are either preexisting within the site or constructed for it, such as a band shelter in the entertainment area or a sandbox in the playground. An object is *major* if it is costly (e.g., band shelter) or is of particular functional significance to its area (e.g., a sandbox), otherwise it is *minor.*

Major object classes are determined by their use areas. For a playground they include swings, sliding boards, sandboxes, and climbing facilities; for a sports field, baseball diamonds, soccer/rugby/football fields, tennis courts, basketball courts, and skating rinks; for water-related activities, ponds, boat-houses, swimming pools, streams, and bridges; for entertainment, band shelters, performance stages, audience areas, merry-go-rounds, and zoos; for food service, restaurants and kiosks; for maintenance, large equipment storage, equipment repair facilities, and administration buildings; for container networks, branches; and for any use area, parking.

A major object to be placed within a design space specifies a particular use area in which it may be anchored; a minor object may or may not do so. Each object has a left-right and a top-bottom dimension, and may also specify (or inherit as defaults) a footprint, scale (relative to the site and/or the other objects present), orientation to natural forces (e.g., sun, wind, precipitation), and cost (as a function of size and material). An object may either be specified with precise property values (e.g., a rectangular 100 by 60 foot tennis court) or by a range of values (e.g., a 6000 square-foot convex object with a height-to-width ratio in [2, 3]). Like a use area, an object may have a refinement function. (See Section 4.1 for details.)

Once again, container networks require special treatment. In addition to its skeleton, each container network has objects called *branches* (major connection objects) from its skeleton to other objects. Branches must connect to the network, and serve to elaborate it. A branch is specified by its network and the object or objects it must connect. A point where a branch joins the skeleton or terminates at an object is called a *node.* If the object is a required branch in a container network, it will also have endpoints and a capacity (e.g., a two-lane road from a parking lot to the periphery). Thus each network is represented by an attribute graph on nodes that records distances and capacities.

For each use area in a sketch, the second pass of pragmatic design synthesis creates a use area plan that anchors each major object in the site, retaining the borders of the use areas as fiat boundaries [20]. For the objects of Table 1, plus branches of the road network from the skeleton to the northern and western edges of the site, one possible plan from the sketch of Fig. 2 is Fig. 3.

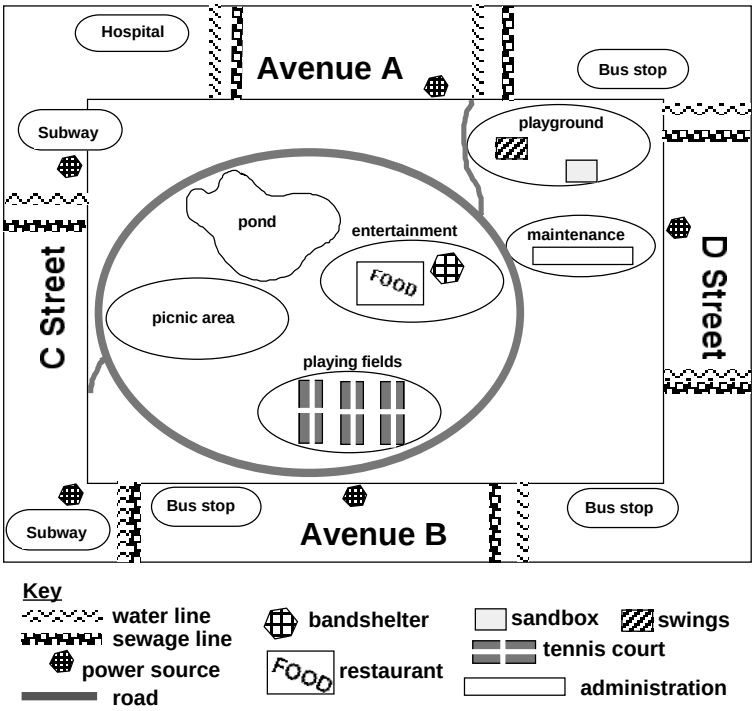

Fig. 3. A plan for the sketch of Fig. 2

3.3 The Design

The third and final pass, at the finest granularity level, has two stages. In its first stage, it refines a plan by positioning all the minor items. The result is a set of computer-originated *designs*. Each object class has an *owner*, an agent realized by a set of design procedures specialized for that class. Minor objects may be anchored by their owners anywhere in the site that satisfies the constraints. (There is a general procedure to do this for any minor object not associated with a use area.) Because minor objects are expected to be small and have few restrictions, often any random location in the vicinity suffices. If the requirements for the running example included seven benches (three allocated to the playground, and two to the playing fields area) and three storage sheds, one possible design for the plan of Fig. 3 appears in Fig. 4.

The function of the principles, the non-required design criteria, is to evaluate design alternatives. Every criterion has an *evaluation function* that measures the degree to which any state meets its particular specification. A constraint's evaluation function is boolean — the state either meets it or does not. A principle's evaluation function, however, is numeric; it may also be complex and make heavy computational demands.

After all plans are completed, the second stage of the third pass *tweaks* them, that is, it re-anchors objects in an attempt to improve the scores received from the evaluation functions. Tweaking is directed by the principles; so, for example, if "buildings should

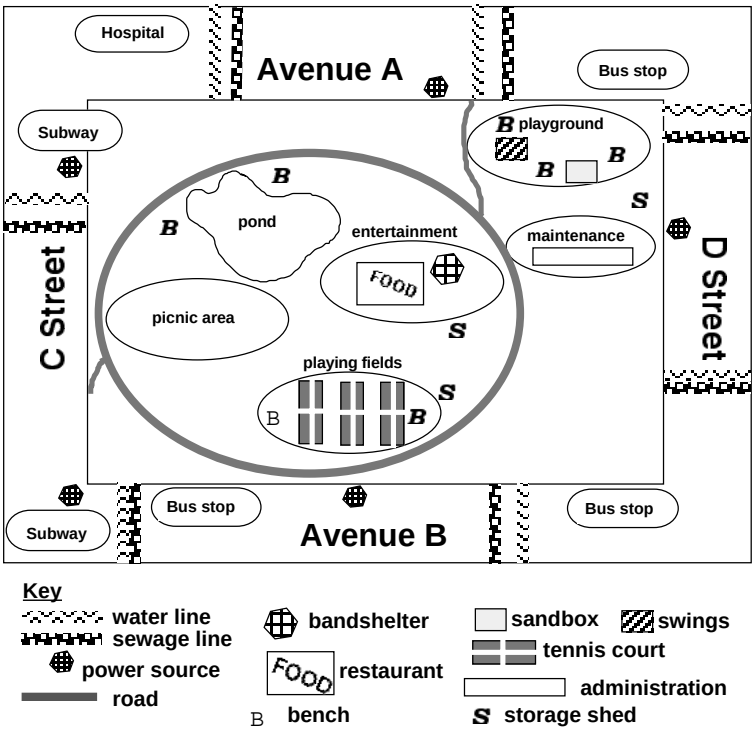

Fig. 4. A design for the plan of Fig. 3

be close together" is a principle, the system may shift one building toward another in a plan, re-rate the resultant plan, and choose the higher-scoring of the two. In empirical testing on simple problems [5], such tweaking was able to raise the best design's score by as much as 30%. Principles that are particularly costly to evaluate may be *reserved*, so that they are used only in initial formulations (of a sketch or a plan) and in tweaking, but excluded from anchoring iterations.

4 Discussion

4.1 Geographic Information for Design

Classification is, to our minds, the essential geographic knowledge for design: it links objects with their use areas and permits design to focus on fewer objects and the constraints relevant to them. Classification also supports the formulation of three kinds of criteria: those that pertain to a single object (*object* specifications), those for a set of objects in the same use area (*intra-area* specifications), and those for a set of objects in more than one use area (*inter-area* specifications). The object specifications derive from the survey data; the intra-area specifications are based on both survey data and

expert knowledge. Inter-area specifications, however, arise primarily from the needs people have within the park. For example, people may require efficient travel, aesthetic views, or informative views. The park management, meanwhile, may require efficient storage and retrieval of equipment, energy conservation, freedom of movement for vehicles, and control for safety.

Once objects are classified, the initial states for both the sketches and the use area plans rely heavily upon knowledge infusion. Even the anchoring of half a dozen use areas to form a sketch can prove computationally expensive. As noted above, the survey phase of park design includes an inventory of pre-existing site features, used by landscape architects to restrict the number of possible design solutions. Landscape architects typically collect terrain data on features such as drainage, slope, soil types, and vegetation, and represent each set of measurements as a *site map*, a separate graphical model. People use site maps to help locate use areas [1]. Each site map can be thought of as a separate, fine-grained grid, where each grid cell represents the value of the corresponding measurement. For example, if slope measurements were categorized into five ranges, each grid cell would be assigned a value corresponding to one of the five ranges, based upon the survey data.

In park design, the nature of the land within the site (e.g., slopes, drainage, soil, and existing vegetation) naturally provides *terrain constraints*. These constraints preclude many possible anchorings, and make some of the remaining ones far more attractive than others. One would not, for example, anchor a camping area on marshy ground, or a playground on a steep slope. Table 2 offers a simplified example of how categorized site data qualitatively determine a land *type*, and how land type then dictates construction appropriate to it. (For example, roads require terrain that is at least category III.)

The efficient generation of use area plans is also directly attributable to classification. Given a sketch, each use area is detailed with its major objects, based upon the relevant inter-class constraints, to produce a use area plan. Recall that each object class

Table 2. Categorized site properties characterize land types [1]. Each type is suited to a particular level of construction.

		Type				
Property		**I**	**IIA**	**IIB**	**III**	**IV**
Slope	0-2%	x	x	x	x	x
	2-4%	x	x	x	x	x
	4-10%	x		x	x	x
	10-20%	x				
	20%+	x				
Drainage	Good	x	x	x	x	x
	Fair	x	x	x	x	
	Poor	x				
Soil	Good	x	x	x	x	x
	Fair	x	x	x		
	Poor	x				
Vegetation	Good	x		x		
	Fair	x		x		
	Poor	x	x		x	x

Type	**Construction**	**Usage**
I	None	trails
IIA	Minimal	playing fields
IIB	Minimal	picnicking
III	Moderate	roads, parking
IV	Intensive	all structures

has an owner (Section 3.3). The owner of an object class is responsible for the satisfaction of any intra-class constraints applicable to its objects. Each owner has a case-based library of *ideal frameworks*, use area plans for a set of objects that abide by all constraints, and are rated highly by all the owner's principles, including any reserved ones. The playground owner, for example, has some ideal frameworks for playgrounds with much climbing equipment, and other ideal frameworks with objects geared to very young children. The owner selects and modifies a library plan to fit the set of major objects in the use area under consideration, much the way the sketch is constructed. Each iteration anchors a major object. It is the restricted focus to objects of the same class that makes these searches manageable.

Refinement functions, both for use areas and for objects, also incorporate spatial knowledge into pragmatic design synthesis. The purpose of a refinement function is to define an object better, either when it is anchored or during tweaking. A refinement function might be expected to produce an aesthetically appealing footprint, or to layout the spaces in a parking lot. One refinement function might prevent an artificial pond from deformation into a stream; another might maintain the convexity of a building. If the object is an aggregate (e.g., forest or parking lot), its refinement function will also specify its components (e.g., trees, parking spaces).

Thus geographic knowledge pertinent to layout design includes object categories, terrain constraints, ideal frameworks, and refinement functions. When infused with such knowledge, a program should be able to create a variety of legal sketches efficiently.

4.2 Alternatives as Design Devices

Discussion of one important aspect of the process has been deferred until now: the encouragement of multiple alternatives. Although the process moves from sketch to plan to design, it does not focus on one "best" idea. Instead, like human experts, pragmatic design synthesis attempts to produce *s* different sketches, and *p* plans for each use area. As a result, with *a* use areas there may be as many as sp^a plans from which to produce a design. (All italicized letters here denote user-set search parameters.)

To produce an individual sketch, search begins from an initial state (see Section 4.4) and, on each iteration, re-anchors one or more use areas. At each iteration the current sketch is scored by the (non-reserved) use-area principles specified in the problem. Each of the *b* most appropriate cases serves as the initial state for *c* searches, and each search is iterated *d* times. The final sketch in each search is retained only if it achieves a sufficiently high score. Once *s* sketches are produced this way, or there have been *bc* searches, the first pass ends.

In the second pass, a set of use area plans for each use area is generated in a similar manner. Every owner has a *recommendation function* that generates *proposals*, suggestions on how to anchor one of its objects in a state. A recommendation function respects the properties of an object or use area (e.g., orientation) and can be tailored to an object's class. For example, it may be appropriate to lop off the corner of a forest or bend a road around an obstacle, but neither of those would be appropriate for a tennis court. To produce a use area plan, search begins from any of *e* good matches for the current use area from among the owner's ideal frameworks. Search iterates until *p* use

area plans are produced for each use area. An iteration entertains proposals from the owner of the use area to anchor (or re-anchor) a major object there. The (non-reserved) inter-area principles score the use area plan that would result from each proposal, and the *f* with the highest scores are used to create (full) plans. A preliminary plan is a combination of high-scoring use area plans. Each preliminary plan is then scored by the principles, and the best of them are forwarded as plans.

In the third pass, each plan is detailed into a design, with anchorings proposed by the owners of the minor objects. Rather than iterate on these inexpensive decisions, the minor objects are distributed into each plan *g* times, and the *h* highest-scoring designs are selected. Every selected design is then tweaked, and the *i* best of those are output from the system.

In the event that there are not enough appropriate matches to provide initial search states, cases may be generated at random, and the highest-scoring among them substituted for historical knowledge. Indeed, in the 13-object problem of Section 2 there were no cases, and randomly-generated initial states served quite well, as long as 25-30% of all computation time was devoted to their production and evaluation [5]. In any event, some randomly-generated cases may be desirable for novelty.

4.3 Navigability and Other Principles

Most principles are intended to capture aesthetic considerations. Once a use area or an object is anchored, one can determine the degree to which it is remote from a particular location (both metrically and, with some qualitative computation, visually). Similarly, once all the objects are anchored, one can also determine the degree to which a particular object is visually connected to other objects, visually contiguous with other objects, or visually isolated from other objects. At that point, for the entire design, one can evaluate the degree to which the objects together display massing, spacing, setback, unity, contrast, and *continuity* (clarity of form and closure). Metrics for all of these are currently under development.

4.4 Implementation Status

Portions of the theory described here have been implemented, either in a pioneering system for urban parks on an empty grid [5] or in a system currently under development for recreational park design [21]. The two have tested many of the ideas discussed here; this section provides a status report on the latter.

For the first pass, the use area sketch, our system is nearly complete. At this writing, however, use areas are represented by circles (whose diameters may be manually altered during search) rather than ovals, and container networks are not yet available. The user specifies a park problem (or *project*) as a set of use areas (with kinds and sizes) and proximity constraints. For example, Fig. 5 displays a sketch for a recreational park with five use areas: two picnic areas, two camping areas, and a playground.

Two kinds of problem constraints on use areas are portrayed in Fig. 5: proximity constraints and terrain constraints. A *proximity constraint* specifies a bound on the distance between two use areas [1]. A proximity constraint is represented in our system

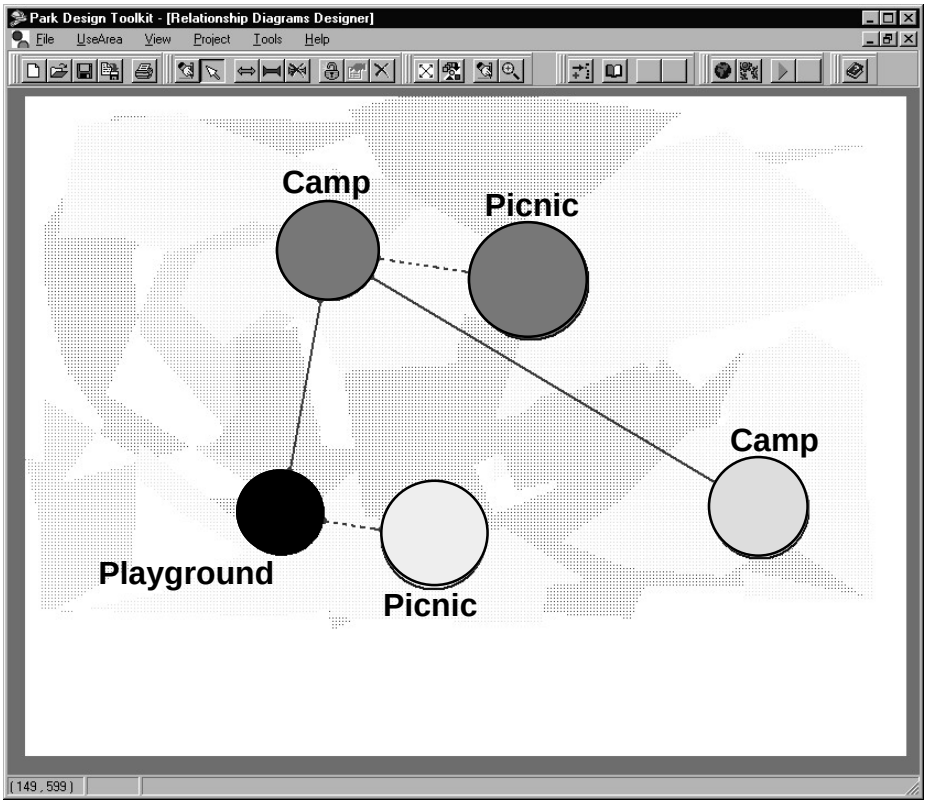

Fig. 5. An implementation-produced sketch. Each circle represents a use area. Proximity constraints appear as lines — dotted lines for attraction constraints, and solid lines for repulsion constraints. Each circle's shading represents the suitability of its current location; black for rejection, deep gray for acceptance, paler grays for moderate acceptance. The background shading is the acceptance map for the playground. Shaded circles and labels have been superimposed here for legibility.

as a link between the circumferences of two use areas. A proximity constraint is either an *attraction constraint* (represented here by a dotted line), which keeps two use areas at most a given distance from each other, or a *repulsion constraint* (represented here by a solid line), which keeps two use areas at least a given distance from each other. For example, Fig. 5 includes an attraction constraint between one camping area and a picnic area, and a repulsion constraint between the same camping area and the playground.

Our system employs Intergraph's Geomedia geographic information system [22] to program a module that supports user-specification of the site maps as bitmaps for a park design project. For each kind of use area in the project, our system uses this database to compute an *acceptance map*, which records how appropriate each grid cell is for such a use area. Instead of Table 2, our terrain constraints are based upon a database that includes the far more elaborate tables of the National Soil Survey [19]. Acceptance map calculations integrate all available measurements, including drainage, slope, soil

types, and vegetation. Fig. 5 is a gray-scale version of an acceptance map, displayed on our system as a set of colored grid cells. The color (in Fig. 5, the shading) of each grid cell reflects its level of suitability for that use area: red for rejection, yellow for moderate acceptance, green for full acceptance, or white when no data is available. (These appear in Fig. 5 as black, paler grays, deep gray, and white, respectively.) When a use area in a sketch is highlighted, our system retrieves and displays the acceptance map for that particular kind of use area; the color of the use area's circle represents the mean value of all the circle's grid cells. In Fig. 5, for example, the playground use area is highlighted, so the background displays the playground acceptance map, and the shading of the playground circle indicates that its current location is unsuitable.

The construction of a sketch begins either with a case from the (terrain-independent, positive) cases in the sketch library or with a sketch provided by the user through a graphical user interface. The sketch case base currently contains 14 predefined sketches obtained from existing park designs, whose use areas and proximity constraints were identified by professional designers. The similarity of the current park project to any stored case is based upon the number of instances of each use area type.

A solution to the sketch design problem anchors the use areas, so that they abide by both the proximity constraints and the terrain constraints. The system can search for solutions autonomously, or the user can move use areas about on the screen with the mouse. During autonomous search, heuristics seek solutions that agree with both proximity constraints and terrain constraints. Attention is directed first to the most constrained use areas within favorable portions of their corresponding acceptance maps. (Space limitations prevent further algorithmic detail here.)

Each iteration re-anchors one or more use areas. Whether the search is automated or user-directed, the terrain constraints color the use areas as their locations improve or degrade, while the proximity constraints shift and pivot the cluster of circles about their links. Left to its own devices, our system searches for solutions and displays several candidate sketches that abide by the proximity constraints and best conform to terrain constraints. The user can choose any such sketch and modify it at will.

Implementation of the second and third passes is current work. Meanwhile, we report here upon an agent-based simulation that computes an important principle: the *navigability* of a design, that is, the ease with which visitors can access particular locations the park. Potential destinations are marked on the sketch as *interest points*. The sketch is annotated with *access points* (ways in and out, e.g., doors or gates) for individual destinations, buildings, and use areas, and for the park itself. Through any of a set of access points, our system can currently introduce hundreds of simple, reactive agents into a design. Each agent is a member of a class (e.g., senior citizen, parent, sports enthusiast) whose profile determines the agent's behavior, including speed, intended destination(s), duration of visit, interests, and willingness to stay on the paths. Nonetheless, each agent is an individual, whose behavior is randomized within the ranges specified by that profile.

In real-time on a video display, our simulation can move several hundred agents from various classes through the design's path network toward their intended destinations, gathering information about crowding and chosen paths. At every time step, each agent responds to the current state. On exit from the park, each agent scores its satisfaction with its visit, based upon its ability to reach its intended destinations, crowding, and any other criteria the user chooses to measure. The simulation's cost is a function

primarily of the complexity of the satisfaction function — more complex criteria may be accommodated by a reduction in the number of individual agents. For now, the navigation simulation serves as a reserved principle, but it also provides data that will soon be used to generate proposals to tweak the design.

Fig. 6 is a snapshot from an agent-based simulation in process. Use areas appear as ovals linked by paths; points of interest appear as large black dots. (Access points are not shown.) The small shaded dots in use areas and on paths represent agents from various classes moving about in the park space. The rectangle at the lower left represents a building in which there is an interest point. One can run multiple simulations to obtain information about navigability of the space, including agents' freedom to move along paths, identification of places subject to crowding, and so on. Our agent-based simulation is not limited to parks; it is applicable to any two-dimensional layout.

4.5 Limitations and Proposed Extensions

Although the theory described here is for recreational parks, it could be extended in a variety of ways. Pragmatic design synthesis is not restricted to parks — it could be extended to planning a town, for example, where use areas would be schools and shopping centers and residential areas, probably with one or two additional levels of granularity dictated by the complexity of the objects. Moreover, although it considers slope

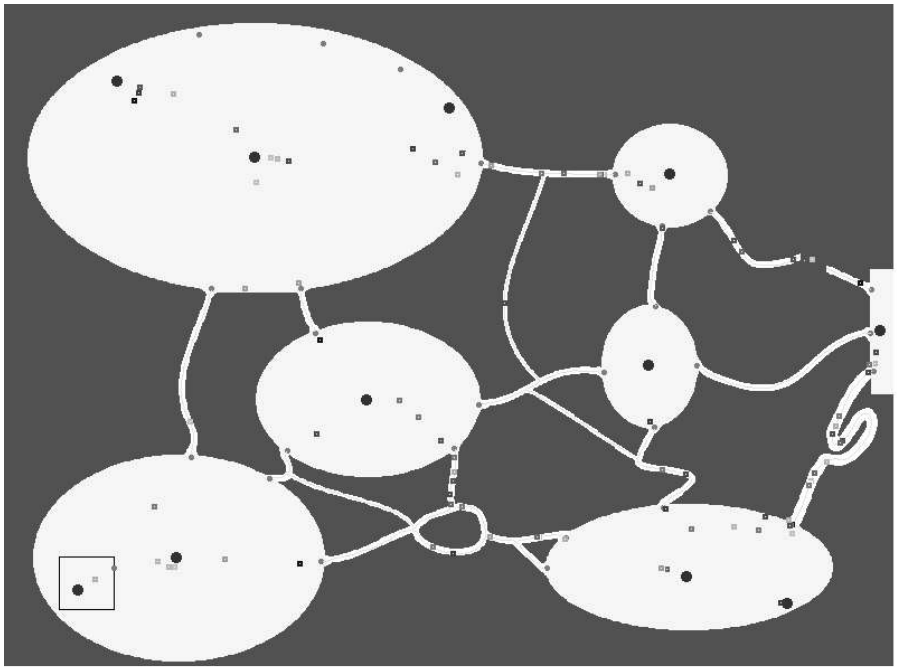

Fig. 6. An agent-based simulation of park usage. Use areas appear as ovals, linked by paths. Points of interest are large black dots; one in the lower left is within a building. Each smaller dot represents an agent; similarly shaded dots are agents of the same class. The park's entry and exit point is at the center of the right margin

in the site maps, pragmatic design synthesis is thus far formulated for two-dimensional space. Although we are well aware of the complexity of aesthetic criteria in three dimensions, we see no other reason to prevent an extension that would include height. We remain convinced, however, that bootstrapping from the soil data narrows search in very practical ways. A set of flat, paved city blocks as a design frame would therefore be more difficult – a clean slate is not an easy tablet on which to design.

One interesting extension would be to have the system extract cases for sketches and use area plans from the most successful (highest scoring) of its own solutions. In this manner pragmatic design synthesis could learn terrain-independent principles from its own experience. For two-dimensional layout design it could prove an effective technique.

Our current approach incorporates case-based reasoning to retrieve initial designs which are then modified with a knowledge-driven search strategy. Future work may involve further applications of case-based reasoning. Case adaptation could be applied in tweaking. Moreover, if the case base contains instances of design decisions that were not ideal, case-based reasoning could be applied during evaluation to ensure that inappropriate design decisions were not repeated.

5 Conclusions

Pragmatic design synthesis produces a variety of constraint-abiding, highly-rated designs for a variety of reasons. Object classification reduces the search space by focusing attention on fewer objects, or on use areas instead of objects. Multiple alternatives provide a broad variety of possibilities. Knowledge informs search: the case bases provide a wealth of historical experience that supports good initial search states, and the recommendation functions encode expert variations. Expensive computations (i.e., reserved principles) are deferred until a design is fully specified. In summary, pragmatic design synthesis is a form of restart hill-climbing from good initial states at different levels of granularity, one calculated to make search both manageable and productive.

References

1. Molnar, D.J., Rutledge, A.J.: *Anatomy of a Park*. McGraw Hill, New York (1986)
2. Goel, V., Pirolli, P., Motivating the Notion of Generic Design within Information Processing Theory: The Design Problem Space. *AI Magazine*. 10 (1989) 19-36
3. Goel, V., Pirolli, P., The Structure of Design Problem Spaces. *Cognitive Science*. 16 (1992) 395-429
4. Schraagen, J.M., How Experts Solve a Novel Problem in Experimental Design. *Cognitive Science*. 17 (1993) 285-309
5. Epstein, S.L. Toward Design as Collaboration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, WI: AAAI. (1998) 135-142
6. Alexander, C., Ishikawa, S., Silverstein, M.: *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York (1977)
7. Alexander, C.: *A New Theory of Urban Design*. Oxford University Press (1987)

8. Cullen, G.: *Townscape*. Reinhold, New York (1968)
9. Stephen-Cutler, L., Stephen-Cutler, S.: *Recycling Cities for People - The Urban Design Process*. Van Nostrand Reinhold, New York (1983)
10. Gottdiener, M., Lagopoulos, A., (eds.): *The City and the Sign: An Introduction to Urban Semiotics*. Columbia University Press, New York (1986)
11. Hedman, R., Jaszewski, A.: *Fundamentals of Urban Design*. Planners Press, Washington, D.C. (1984)
12. Lynch, K.: *What Time is This Place?* MIT Press, Cambridge, MA (1971)
13. Lynch, K.: *Good City Form*. MIT Press, Cambridge, MA (1981)
14. Moughtin, C.: *Urban Design: Green Dimensions*. Butterworth-Heinemann, Oxford (1996)
15. Peps, J.W.: *1794-1918: An International Anthology of Articles, Conference Papers, and Reports selected, Edited, and Provided with Headnotes*. (2000) Cornell University
16. Navinchandra, D.: *Exploration and Innovation in Design: Towards a Computational Model*. Springer Verlag, New York (1991)
17. Domeshek, E., Kolodner, J.: A Case-Based Design Aid for Architecture. In: J. Gero (ed.) *Artificial Intelligence and Design.* Kluwer Academic, Boston (1992)
18. Barber, J., Bhatta, S., Goel, A., Jacobsen, M., Pearce, M., Penberthy, L., Shankar, M., Stroulia, E.: AskJef: Integrated Case-Based Reasoning and Multimedia Technologies for Interface Design Support. In: J. Gero (ed.) *Artificial Intelligence in Design*. Kluwer Academic, Boston (1992)
19. SIRG: *Soil Interpretation Rating Guides, National Soil Survey Handbook (Part 620)*. (2001)
20. Smith, B., Varzi, A., Fiat and Bona Fide Boundaries. *Philosophy and Phenomenological Research*. To appear (2001)
21. Moulin, B., Chacker, W., Epstein, S. Preliminary Design of a Software Tool for the Design of Geographic Space. In *Proceedings of the GEOIDE General Conference 2000*. Calgary, CA. (2000) CD-ROM
22. Intergraph: *Geomedia Professional*, . (2001) Intergraph